(13)

DYNAMIC ANALYSIS OF ASTRONAUT MOTIONS DURING EXTRAVEHICULAR ACTIVITY

by

Grant Schaffner

SUBMITTED TO THE DEPARTMENT OF AERONAUTICS AND ASTRONAUTICS IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 1995

Signature of Author _____
Department of Aeronautics and Astronautics
May 18, 1995

Certified by _____
Professor Dava J. Newman
Thesis Supervisor

Accepted by _____
Professor Harold Y. Wachman
Chairman, Department Graduate Committee

DYNAMIC ANALYSIS OF ASTRONAUT MOTIONS DURING EXTRAVEHICULAR ACTIVITY

by

Grant Schaffner

Submitted to the Department of Aeronautics and Astronautics on May 22, 1995 in partial fulfillment of the requirements for the Degree of Master of Science in Aeronautics and Astronautics.

Abstract

Simulation of astronaut motions during extravehicular activity (EVA) tasks was performed using computational multibody dynamics methods. The application of computational dynamic simulation to EVA was prompted by the realization that physical microgravity simulators have inherent limitations: viscosity in neutral buoyancy tanks; friction in air bearing floors; short duration for parabolic aircraft; and inertia and friction in suspension mechanisms. These limitations can mask critical dynamic effects that later cause problems during actual EVAs performed in space.

Methods of formulating dynamic equations of motion for multibody systems are discussed with emphasis on Kane's method, which forms the basis of the simulations presented herein. Formulation of the equations of motion for a two degree of freedom arm is presented as an explicit example. The four basic steps in creating the computational simulations were: *system description*, in which the geometry, mass properties, and interconnection of system bodies are input to the computer; *equation formulation* based on the system description; *inverse kinematics*, in which the angles, velocities, and accelerations of joints are calculated for prescribed motion of the endpoint (hand) of the arm; and *inverse dynamics*, in which joint torques are calculated for a prescribed motion. A graphical animation and data plotting program, EVADS (EVA Dynamics Simulation), was developed and used to analyze the results of the simulations that were performed on a Silicon Graphics Indigo2 computer.

EVA tasks involving manipulation of the Spartan 204 free flying astronomy payload, as performed during Space Shuttle mission STS-63 (February 1995), served as the subject for two dynamic simulations. An EVA crewmember was modeled as a seven segment system with an eighth segment representing the massive payload attached to the hand. For both simulations, the initial configuration of the lower body (trunk, upper leg, and lower leg) was a neutral microgravity posture. In the first simulation, the payload was manipulated around a circular trajectory of 0.15 m radius in 10 seconds. It was found that the wrist joint theoretically exceeded its ulnal deviation limit by as much as 49.8° and was required to exert torques as high as 26 N-m to accomplish the task, well in excess of the wrist physiological limit of 12 N-m. The largest torque in the first simulation, 52 N-m, occurred in the ankle joint. To avoid these problems, the second simulation placed the arm in a more comfortable initial position and the radius and speed of the circular trajectory were reduced by half. As a result, the joint angles and torques were reduced to values well within their physiological limits. In particular, the maximum wrist torque for the second simulation was only 3 N-m and the maximum ankle torque was only 6 N-m.

Thesis Supervisor: Dr. Dava J. Newman

Title: Assistant Professor of Aeronautics and Astronautics

Biographical Note

Grant Schaffner was born in Nigel, South Africa on 31 December 1965. At present, he is a permanent resident of the United States. He received his Bachelor of Science degree from the Massachusetts Institute of Technology in June 1989. Following this, he worked at Payload Systems Inc. in Cambridge, Massachusetts for two years on hardware design and development for spaceflight experiments performed on the Space Shuttle and Mir space station. Before returning to MIT to pursue graduate study, he spent two years at Space Industries, Inc., in League City, Texas where he worked on the Wakeshield Facility, COMET spacecraft, and advance engineering studies. He is currently studying jointly in the MIT Department of Aeronautics and Astronautics and the Harvard-MIT Medical Engineering Medical Physics Program. He is a member of the ΣΓΤ, ΤΒΠ, and ΣΞ honor societies.

Acknowledgements

This research was sponsored by NASA grant NAGW-4336
and NASA contract NAS1-18690.

I would like to thank Professor Dava Newman for her guidance and kindness over the past two years and for the many hours she devoted to editing this thesis. Her enthusiasm and sense of humor provide much encouragement, particularly after an all-nighter. Much thanks to David Rahn for all the effort he put into developing EVADS and for all his computer graphics wizardry, especially in light of his heavy course load. Steve Robinson, now living out his dream of becoming an astronaut, was a tremendous help in performing the Intelsat VI simulation. Also, much appreciation goes to Dr. Chuck Oman and Professor Larry Young for allowing me to be part of the MIT Man-Vehicle laboratory.

Mike Sherman and Dan Rosenthal at Symbolic Dynamics, Inc. were lifesavers. Without their assistance in learning to use SD/FAST, and their great patience during many telephone calls, the Spartan simulations could not have been done.

There are many people at NASA that I would like to thank: Cab Callaway for his enthusiasm, his belief in our research goals, and his assistance in finding funding; Dr. Harry Holloway for his generous support; Jim Maida and Abilash Pandya for the use of the NASA human strength model; Cal Seaman for EVA data and much other help; Dr. Jeff Hoffman for his encouragement and ideas; and Jerry Miller for his support.

At the MIT Artificial Intelligence Laboratory, I wish to thank Prof. Raibert, Rob Playter, and Robert Ringrose who kindly allowed us to use their *Creature Library* program to perform the Intelsat dynamic simulation.

At Musculographics, Inc., thanks go to Dwight Meglan, for the copy of his PhD dissertation and other assistance during the early stages of my research, and Peter Loan for granting us the rights to use SIMM as a beta test site.

To all my friends, thanks for making these two years as enjoyable as they were productive. In particular, I must thank, Zsuzsanna, for rescuing my sanity during the doctoral qualifying exams, and Karl and Jennifer for providing much comic relief during the final exhausting weeks of thesis completion. Also Dave, Jeni, Keoki, Steve, Lois, Scott, Allison, Nikolai, Rob, Jason, Yanni, Jurgen, Laura, Matt, Nikol, Amy, Liz, my many new ISU friends, and all the highly skilled soccer players of the MIT Graduate Soccer Club.

Finally, a great deal of thanks goes to my mother and father for their unfailing support and encouragement over the years and over the distance, and to my whole family for the warm welcomes received during the all too seldom trips back home to South Africa.

Freedom Day, South Africa
28 April 1994

I have fought against
white domination,
and I have fought against
black domination.
I have cherished the ideal of
a democratic and free society
in which all persons live
together in harmony and
with equal opportunities.
It is an ideal which I hope
to live for and to achieve.
But if needs be, it is
an ideal for which I am
prepared to die.

-Nelson Mandela, spoken from the dock in 1964, before being sentenced to life imprisonment on Robben Island.

Never, never, and never again
shall it be that this beautiful land
will again experience
the oppression of one by another...

-Nelson Mandela, during his South African Presidential Inauguration speech. May 10, 1994.

Table of Contents

List of Figures

List of Tables

List of Acronyms and Abbreviations

c.m.	center of mass
CAD	Computer Aided Design
CAT	Computerized Axial Tomography
d.o.f.	degree(s) of freedom
EMU	Extravehicular Mobility Unit (Shuttle Spacesuit)
ESA	European Space Agency
EVA	Extravehicular Activity
EVADS	EVA Dynamic Simulation
g	acceleration of free fall on Earth (9.806 m/s^2)
JSC	Johnson Space Center
LCVG	Liquid Cooling and Ventilation Garment
LEO	Low Earth Orbit
MIT	Massachusetts Institute of Technology
MMU	Manned Maneuvering Unit
MRI	Magnetic Resonance Imaging
NASA	National Aeronautics and Space Administration
OOM	Object Oriented Modeler
ORU	Orbital Replacement Unit
PFR	Portable Foot Restraint
PLSS	Portable Life Support System
RMS	Remote Manipulator System
rpm	revolution(s) per minute
SSM	Solid Surface Modeler
STS	Space Transportation System
TMG	Thermal and Micrometeoroid protection Garment
WET-F	Weightless Environment Training Facility

1. Introduction

By way of introduction, this first chapter familiarizes the reader with the concept of extravehicular activity performed in spaceflight, the motivation for applying multibody dynamical simulation to extravehicular activity, and the objectives and contribution of the research effort. The chapter concludes with a synopsis of the contents of the thesis.

1.1 Description of Extravehicular Activity

Extravehicular activity (or EVA) is a term commonly used in spaceflight operations to refer to the performance of tasks by an astronaut outside the confines of the space vehicle that serves as his normal habitation volume. Actually, the use of the word "extravehicular" is somewhat of a misnomer since, for most conceivable EVA tasks performed in spaceflight, the astronaut must wear a spacesuit which provides all the necessary life support functions that the home spacecraft provides. Thus, in essence, the astronaut's spacesuit is a miniature spacecraft. Generally, to perform an EVA, an astronaut is provided with an individual life support system and spacesuit that allow him sufficient dexterity and tactility to physically interact with objects in the space environment. Due to the harshness of the space environment, and the multiple functions that the spacesuit must perform, the spacesuit places significant constraints on the astronaut's movements and sensing ability. This, compounded with other factors such as weightlessness, vacuum, thermal loads, and disorientation, makes the performance of tasks during EVA very challenging.

The first EVA was performed by Alexei Leonov of Russia in 1965, followed shortly thereafter by the first American spacewalk, performed by Edward White. The purpose of these early EVAs was simply to prove the feasibility of placing humans in free space (Newman and Barratt 1995). EVA became an operational capability during the Apollo program and for the first time humans were able to walk on another celestial body. During the Russian Salyut and Mir space station programs, and during the U.S. Space Shuttle program, EVA proved its worth as a highly versatile operational capability. EVA should be looked upon as a valuable resource for space station construction and future human space exploration efforts.

Recently, the EVAs performed on Mir and the Space Shuttle Orbiter have become more complex and more challenging. Good examples are satellite capture and repair missions, the Hubble Space Telescope repair mission, and contingency EVAs. During

some of these missions the dynamics of the environment have proven to be counterintuitive during the execution of EVA tasks and in some cases have led to difficulties and even failures. For instance, when an astronaut attempted to capture a slowly spinning Intelsat VI satellite with a retrieval mechanism during the shuttle mission STS-49, a lack of appreciation of the dynamics of the system, and limitations of the physical simulators used to train for the mission, led to repeated failures of the satellite capture procedure. The astronauts and mission control were forced to find an alternative solution (Holloway 1992). Crewmembers, performing the first three person EVA in history, positioned themselves at three points within the payload bay of the Orbiter and grabbed the base of the satellite with their gloved hands. It was an impressive and inspiring performance, albeit risky and unplanned.

Until now, development and training activities associated with planned EVAs, in both Russian and U.S. space programs have relied almost exclusively on physical simulators. These include neutral buoyancy facilities, air bearing floors, aircraft flying Keplerian trajectories, and suspension systems. Each facility, however, has inherent limitations that prevent true simulation of the EVA environment. In the water tanks it is hydrodynamic drag; on the air bearing floors it is friction and limited degrees of freedom; in parabolic aircraft it is limited duration; and in the suspension systems it is limited degrees of freedom and range of effectiveness. Using a combination of these simulators tends to compensate for their limitations, but still cannot account for every effect of weightlessness.

1.2 Motivation

To complement and make up for the shortcomings of physical simulators, an effort to develop a computational simulation system to model human dynamic motion has been initiated. Although the system's utility extends to a broad range of human motion tasks and robotics, including earth-based studies, it is being developed with the prime objective of producing a valuable analysis tool for evaluating the performance of human motion in weightlessness during extravehicular activity. This research effort takes advantage of experience gained in the field of computational simulation of multibody dynamics.

Various techniques have been developed to obtain the dynamics equations of a multibody system. The most well known methods include the classic Newton-Euler approach, the Lagrangian formulation, and Kane's method. There are two principal approaches to the formulation of the dynamic system equations: symbolic (or algebraic) and numerical. The symbolic approach uses one of the three methods given above to derive the dynamic equations, either by hand (which becomes impractical for systems of

more than three bodies) or using a computer program. Two examples of such programs are AUTOLEV[1] (Schaechter and Levinson 1988) and SD/FAST[2] (Hollars, Rosenthal et al. 1994), both of which are based upon Kane's method (Kane and Levinson 1985). There are also programs available for generating the equations using numerical approaches. Some examples include DADS (Haug 1989) and ADAMS (Chace 1978). In addition, commercial software developed recently has introduced powerful means of analyzing quasi-static motions of human musculoskeletal systems, for example SIMM[3] .

It is a relatively new idea to apply computational techniques to simulation of EVA tasks. Some existing programs have focused on EVA simulation as their prime objective, but most of these programs are aimed at either computer graphics and CAD (Price, Fruhwirth et al. 1994) or anthropometric applications (McDonnell-Douglas 1994). Few, if any, of these programs go into any significant detail on the simulation of multibody dynamics.

An important advantage of multibody dynamic simulation is the ability to account for effects that are absent in physical simulators or masked by their limitations. In addition, this type of simulation can be performed relatively economically and quickly. It represents a particularly attractive capability when used as a complement to the existing simulators.

Some disadvantages of computational simulation are the rapid rise in complexity as the number of bodies and degrees of freedom in the system increase, and the occurrence of multiple solutions to inverse kinematic and inverse dynamic analyses in systems with redundant degrees of freedom (Asada and Slotine 1986). These difficulties can be minimized by keeping the model as simple as possible, and by using constraints or optimization to select a particular solution. Another issue to be considered is the accuracy with which physiological systems of the human are modeled. Mass properties of human body segments and joint kinematics are usually highly simplified in these types of analyses and it is wise to compare the data obtained from simulation with experimental results.

It is clear that a need exists for a method of EVA simulation that avoids the shortcomings of physical simulators and which is at the same time economical and convenient to use. Describing how this need can be addressed is the subject of this thesis.

---

[1]OnLine Dynamics, Inc., Sunnyvale, CA

[2]Symbolic Dynamics, Inc., Mountain View, CA.

[3]Musculographics, Inc., Evanston, IL.

1.3 Objectives and Contribution

The primary objective of the research described in this thesis is to demonstrate the significant value of multibody dynamics analysis for the simulation of EVA tasks. In particular, it is believed that computational dynamic simulation has certain advantages over physical simulators and complements them very well. The main advantage is the ability to represent all the degrees of freedom available to a body in weightlessness (six degrees of freedom for a single isolated body - three translational and three rotational) and at the same time avoid some of the detrimental effects that physical simulators are subject to, such as friction in air bearing floors or viscous drag in neutral buoyancy facilities. Other advantages include low cost, flexibility, quick turnaround, low manpower requirements, and ease of operation.

Of course there is a price to pay for all these advantages. Simulation of multibody dynamics is inherently computationally intensive. In addition, greater accuracy and realism is gained through adding degrees of freedom and geometric detail to the dynamic system model, further increasing the complexity of the simulation, the computer processing time, and the work required of the analyst. In practice, the analyst must consider a tradeoff between the complexity of the dynamic system model and the accuracy and validity of the results in regards to the particular situation being simulated.

Certain specific objectives were established to guide the research effort. These 8 prioritized objectives are:

1) Develop a convenient means of modeling the dynamic system and tailor it to the particular needs of EVA simulation.

2) Transform the description of the dynamic system into equations of motion represented in computational form.

3) Develop computer code to drive simulations of the dynamic system under a variety of conditions.

4) Explore methods of prescribing the motions to be performed in a task-oriented form, the way that an astronaut or trainer might think of the operation, without the need to explicitly specify the kinematics (positions, velocities, and accelerations) of each segment. In other words, perform an inverse kinematics analysis, given only the motion of the endpoint of the system.

5) Determine the joint torques required to drive the system in performing a particular motion by using the calculated segment kinematics in an inverse dynamics analysis.

6) Create a graphical animation and data display user interface.

7) Show how the results of the inverse kinematics and inverse dynamics analyses are interpreted.

8) Demonstrate the means by which simulations are improved in an evolutionary manner.

To achieve these objectives a seven segment model of an astronaut is created with an eighth segment attached to the hand representing a large mass to be manipulated during an EVA task. The equations of motion are derived using SD/FAST, based on Kane's method, which creates code to represent the equations of motion in computational form. Additional code is created to drive the system during two simulations, the second simulation being an improvement on the first. Finally, the results are visualized through the aid of an animation and data plotting program, which has been named EVADS (EVA Dynamic Simulation). All of these operations are performed on a Silicon Graphics Indigo2 computer.

1.4 Synopsis of Thesis

In addition to the introduction chapter presented here, this thesis is comprised of chapters dealing with background information, methodology, results, and discussion and conclusions. The Background chapter provides a brief history of EVA; a description of the space environment encountered in EVA (mostly in low Earth orbit); a summary of spacesuit requirements and the effects of the spacesuit's construction on human body dynamics; a discussion of training methods and physical simulators used in preparation for EVA; an introduction to computational simulation of multibody dynamics; and an example of dynamical simulation as applied to the modeling of the Intelsat VI satellite capture EVA.

The Methodology chapter first works through an example of dynamical equation formulation applied to a simple two degree of freedom system; then describes the Space Shuttle mission STS-63 Spartan mass handling EVA which serves as the subject for the

two main simulations presented in the thesis; and finally presents the details of how the dynamic simulations are performed, including the creation of the system's description file, formulation of the equations of motion, inverse kinematics, inverse dynamics, and animation and data plots using EVADS. In both simulations, the EVA task is to manipulate the Spartan payload around a circular trajectory. In the first simulation, the radius of the circle is 0.15m and the trajectory is completed in 10 seconds. In the second simulation, the radius of the circle is 0.075m and the trajectory is completed in 20 seconds. In addition, the lower body joints (ankle, knee, and hip) are fixed in the first simulation, but are allowed some compliance (by means of passive springs and dampers) in the second simulation.

The Results chapter first presents data obtained from joint torque test functions, followed by data obtained from the two EVA simulations. It is observed that, in the first simulation, the wrist joint exceeds the range of motion limits and is required to exert torques beyond the level of human capability. The second simulation solves these problems by starting with the arm in a more comfortable position and by requiring the hand to follow a smaller circular trajectory at a lower speed (thus requiring less torque). The chapter concludes by presenting results of the animation and data plotting functions of EVADS.

In the Discussion and Conclusion chapter, the research objectives are recapped and the extent to which they were achieved is indicated. The results of the two simulations are deliberated with particular attention given to the success of implementing compliance in the lower body joints. The effectiveness of the animation and data representation abilities of EVADS is assessed. Some general conclusions about the research are drawn, especially relating to the feasibility of using computational multibody dynamics to simulate EVA tasks. Finally, the chapter concludes with suggestions for further research and a short summary of the thesis.

2. Background

This chapter presents background information on EVA and multi-body dynamic simulation to familiarize the reader with the unique challenges of performing EVA, the manner in which astronauts prepare for spacewalks, and how multibody dynamic simulation can improve this preparation. It begins with a brief synopsis of the EVA experience in both the U.S. and Russian space programs. Owing to the relative brevity of U.S. Space Shuttle flights and the long hiatus due to the Challenger accident, American astronauts are far behind their Russian counterparts in the total number of EVA hours accumulated.

The relative inexperience of American astronauts in performing spacewalks has become an issue in regards to the rigorous EVA assembly schedule planned for the International space station. According to the latest estimates, astronauts and cosmonauts will spend about 888 EVA hours assembling the station. An additional 171 hours per year are expected for maintenance requirements (Harwood 1995). Concern has been raised as to whether NASA will be able to support such a vigorous schedule of spacewalks. NASA has recently expressed confidence that EVA has become a routine operation and that the demands of the space station assembly task can be met. To make sure that EVA crews are not overworked, limits have been designed into the EVA timelining protocol. In general, this ensures that crewmembers will have at least one rest day between consecutive EVAs and will not be called upon to perform an EVA before the fourth day of flight after they have overcome any symptoms of space motion sickness. Nevertheless, it is clear that it will be more important than ever to ensure that astronauts are well prepared to perform these spacewalks and have a good understanding of the environment they will encounter. A major part of this preparation should include both training and analysis - qualitative and quantitative - of the dynamics of object manipulation, tool operation, body translation, and body orientation. All complex tasks should be carefully analyzed. Even tasks that might be perceived as relatively simple could have unexpected results when performed in the deceptive environment of low Earth orbit, a lesson that has been hard learned in past EVAs.

After the following section on EVA history, a brief description of the space environment is presented. Environmental factors determine the life support functions that a space suit must provide, but the physical realities of spacesuit construction have a significant effect on an astronaut's mobility and senses.

Until recently, EVA research and training has been conducted almost exclusively by means of physical simulators. Different types of physical simulators and their strengths and weaknesses are described later in this chapter followed by a discussion of computational simulation methods and dynamic equation formulation in multi-body systems. In particular, various ways in which computational simulation can be applied to EVA are identified.

The chapter concludes by describing a simple demonstration of dynamic simulation applied to an actual EVA. The goal of the spacewalk was to capture a stranded Intelsat satellite and attach it to a new upper stage kick motor. Unfortunately, a lack of understanding of the dynamic environment, compounded by the limitations of physical simulators, led to failure of the rehearsed procedure. The main reasons for this failure are demonstrated by the simulation results.

2.1 A Brief History of EVA

A good overview of the history of extravehicular activity is provided by Newman and Barratt in Chapter 22 of *Space Life Sciences* (Newman and Barratt 1995). The first EVA was performed by Russian cosmonaut Alexei Leonov in March 1965. The EVA lasted for ten minutes, during which Leonov floated free of his Voskhod capsule while attached to a 5 m umbilical which supplied air and communications. Edward White became the first American astronaut to perform an EVA in June 1965 during a Gemini-Titan 4 flight. These EVAs were primarily to demonstrate that a human could operate in free space while protected by a space suit which provided a pressurized gas environment for respiration, air cooling, and prevention of edema and blood boiling. Since the early spacesuits had restricted mobility and lacked active body cooling, the spacewalkers quickly became fatigued while performing even simple tasks. Other difficulties underscored the need for restraint mechanisms such as foot restraints, handholds, and tethers.

EVA became an operational capability during the Apollo program. From the time Neil Armstrong and Edwin "Buzz" Aldrin set foot on the moon, Apollo astronauts accumulated a total of 160 hours of extravehicular activity time on the lunar surface. They traveled 100 kilometers on foot and with the lunar rover, and collected 2,196 rock and soil samples during these highly successful EVAs.

The Apollo spacesuit design exhibited significant improvements over the spacesuits used in the Mercury and Gemini programs. A major modification was the replacement of the life support umbilical with an autonomous backpack life support system which supplied breathing oxygen, pressurization, and body temperature regulation through a

liquid cooling garment. In addition, the suits had much greater mobility than previous generation suits, especially in the legs, allowing for better locomotion.

The value of EVA was dramatically demonstrated during the Skylab program. A portion of the space station's outer skin and one of the solar panels were lost during launch, resulting in an overheated and underpowered station. The mission was salvaged when astronauts Joseph Kerwin and Charles "Pete" Conrad erected solar shades and freed the second solar panel by means of EVA. The temperature of the station was reduced to the nominal range and sufficient power was then available to run the station's systems.

The Russians had meanwhile reduced their usage of EVA, due to difficulties in their early attempts. They resumed EVA during the Salyut 6 program (1977-1981) to replace equipment and retrieve experiment samples from outside the space station. Cosmonaut Georgi Grechko performed an important EVA to determine whether the cone of the Salyut 6 docking port had been damaged. During the Salyut 7 program, experience was gained in construction, telemetry and material science. EVAs were also performed to study cosmic radiation. A significant milestone occurred on July 25, 1984, when cosmonaut Svetlana Savitskaya became the first woman to perform an EVA. She used a portable electron beam device to cut, weld, and solder metal plates.

NASA increased it's EVA experience during the Space Shuttle missions. Spacewalks were divided into two categories: planned and contingency EVAs. Planned EVAs focused on specific mission objectives. Contingency EVAs allowed astronauts to respond to unexpected problems during the actual mission. The thirteen two-person EVAs performed between 1983 and 1985 from the Space Shuttle Orbiter met the goals of: demonstrating and evaluating the Extravehicular Mobility Unit (EMU), a 29.6 (4.3 psi) spacesuit; testing the Manned Maneuvering Unit (MMU); working with the Remote Manipulator System (RMS); assembling space structures; and repairing equipment and satellites. After a five year lapse, largely due to the Challenger accident, EVAs were resumed in the Shuttle program in April, 1991. Since then, the diversity of operations performed during spacewalks has significantly increased. The most recent EVAs have focused on preparations for assembly of the international space station, such as crew translation aids, mass handling, and assembly techniques.

The most recent Russian EVAs have occurred during the Mir space station program. In April 1987, Yuri Romanenko and Alexander Laveikin performed a contingency EVA to remove an obstruction causing a docking problem between the Mir and Kvant modules. In February 1990, the first test of the Russian SPK unit (similar to the American MMU) was performed. In July 1990, cosmonauts Alexander Balandin and Anatoly Solovyov performed an exciting EVA in which they extended a ladder from the Kvant 2

module to the Soyuz capsule to repair some heat insulation blankets. Thanks to their continuous presence in orbit with the Mir space station, in contrast to the short trips into space on the Shuttle in the American space program, the Russians were able to accumulate many more hours of EVA experience than the Americans.

Even though NASA has gained experience in a wide variety of EVA tasks, the actual number of hours of EVA time accumulated is still relatively small. In fact, the number of EVA hours logged to date by US astronauts is only about one quarter of the number of hours estimated for the construction of the space station. In addition, there is still much room for spacesuit improvements, especially in terms of mobility, comfort, operating pressure, and longevity. Of greatest significance to the research effort described in this thesis, is the fact that the complexity of multibody dynamics in space continues to be a significant difficulty for EVA astronauts, as was evident in the STS-49 EVA which is described in a later section. Other than empirical anthropometric studies, little has been done in the way of quantitative analysis and simulation of EVA. The purpose of this thesis is to demonstrate the great value of research efforts in this area.

Before discussing multibody dynamics in more detail, it is worth taking a brief look at the space environment that the EVA crewmember encounters. The realities of this environment dictate that an astronaut must wear a spacesuit when he leaves the confines of his spacecraft. The physical properties of a spacesuit, however, have a significant impact on the astronaut's mobility. Properties of the space environment and spacesuits are described in the next two sections.

2.2 The Space Environment and EVA

The majority of EVAs performed to date, with the exception of the Apollo missions which left Earth orbit, have taken place in the space environment of low Earth orbit (LEO). The LEO environment is characterized by hard vacuum, weightlessness, various forms of radiation, and micrometeorites and orbital debris. A good description of the space environment is provided by Griffin and French in the text *Space Vehicle Design* (Griffin and French 1991).

In a typical low Earth orbit, say at 200 km, the 1976 US Standard atmosphere lists a temperature of 855 °K (582 °C), a pressure of 8.47×10^{-5} N/m^2 (1.22×10^{-8} psi) and a density of 2.54×10^{-10} kg/m^3 (1.59×10^{-11} lb/ft^3). The very high temperature can be attributed to direct solar radiation when unprotected by the atmosphere's shielding. The low pressure and density are evidence of the hard (but not complete) vacuum, even in a relatively low orbit. Outgassing occurs in most materials exposed to this vacuum.

Furthermore, ionized atomic oxygen present at this altitude is highly reactive and may degrade materials.

Weightlessness has a major impact on extravehicular activity. A weightless condition is experienced in orbit because centripetal acceleration gives rise to a centrifugal force which counterbalances the gravitational attraction of the Earth. Small accelerations in the range of 10^{-3} to 10^{-11} -g (a "g" is one standard unit of Earth's gravitational acceleration, a value of 9.806 m/s^2) are nevertheless encountered due to various perturbations (i.e., space gas drag, thruster firings, crew disturbances, etc.). For instance, during the early EVAs performed in the Gemini program it was discovered that an unexpectedly high level of effort was required to perform even simple tasks. During Gemini 9, astronaut Gene Cernan was unable to test a maneuvering backpack (for controlling body translation and orientation) due to exhaustion experienced in putting on the unit (Griffin and French 1991). Other difficulties were encountered by astronauts in simply handling life support tethers and shutting the spacecraft hatch following an EVA. These difficulties were attributed in part to the bulkiness of the spacesuit, but more significantly to the lack of surfaces and masses to provide the reaction forces normally supplied by the combination of friction and gravity on the Earth's surface. A study conducted by Wortz during the Apollo era attempted to determine why human performance capability was sometimes diminished during spacewalks, leading to increased energy expenditure (Wortz and Prescott 1966). Wortz concluded that the lower efficiency and increased metabolic cost of work observed during some activities was due to a combination of the muscular work required to provide necessary reactive forces to compensate for the lack of traction and the physical constraints of the spacesuit. More recently, careful placement of handholds and foot restraints has enabled astronauts to improve their efficiency and perform a wide variety of tasks without exhausting themselves. It should be pointed out that in spite of these difficulties, humans are able to accomplish tasks in weightlessness that could never be accomplished on Earth, such as manipulating very large masses.

Another serious concern for astronauts performing spacewalks is the radiation environment. Almost all of the electromagnetic radiation in the solar system is emitted by the Sun. Near the Earth, the energy density is about 1390 W/m^2. Thermal radiation produces wide fluctuations in temperature in bodies in low Earth orbit. Very high temperatures, in the range of 800 to 900K, are experienced by surfaces exposed to direct solar radiation, whereas surfaces that are in shadow, and exposed only to open space, can experience extremely low temperatures (less than 100K). Ionizing radiation consists of high-energy particles (protons and electrons) and photons. Galactic Cosmic Rays are an additional source of radiation consisting of high energy photons, alpha-particles, and

heavy nuclei. Fortunately, for EVAs occurring in low Earth orbit, much of this radiation is shielded by the Van Allen belts in the geomagnetic field.

Micrometeoroid flux near Earth is calculated on a log scale versus particle mass with few particles exceeding 10^{-6} g. For most purposes, a density of 0.5 g/cm^3 and a velocity of 20 km/s can be used as average values. Most micrometeoroids are extremely small. The rule of thumb is that a 1 μg particle will just penetrate a 0.5 mm thick sheet of aluminum. Space debris flux levels exceed those of micrometeoroids and thus pose a greater threat. It is estimated that there are 5408 objects larger than 10 cm, approximately 40,000 objects larger than 1 cm, and literally billions of particles in the 0.01 to 0.5 mm range in Earth orbit. This fact must be considered when designing spacesuit outer shells to resist puncture by means of particle impact. Of course a tradeoff between particle size and impact probability must be exercised

2.3 Spacesuit Requirements and Implications

It is clear from the description in the previous section that the space environment is not a friendly one for humans. To be able to venture beyond the confines of their spacecraft, astronauts must don spacesuits with associated life support systems (which may be portable or connected via an umbilical). The spacesuit and life support system must provide the human with the necessary requirements for sustaining life and at the same time protect them from the hazards of the environment.

The spacesuit itself must provide three basic requirements: a pressure vessel around the astronaut; protection from environmental hazards; and mobility for performing motion tasks and doing work. The life support system, or portable life support system (PLSS), must supply the following: pressure control and oxygen supply; removal of carbon dioxide and trace contaminants; thermal control; humidity control; and power, communications, and data display. A PLSS has slightly different requirements than a vehicle life support system: it is generally expected to operate for a shorter duration; it does not necessarily need to supply food or water; and it is more closely coupled with the human's metabolic rate (Webbon 1994).

The most basic requirements for a short term life support system (not including water and food) are given in Table 2.1. The two primary categories are air and temperature.

Table 2.1 Basic requirements for a short term life support system.

Air	
Total Pressure	160-170 mmHgA (3.09-14.70 psi)
Oxygen Partial Pressure	160-260 mmHgA (3.09-5.03 psi)
Nitrogen Partial Pressure	0-600 mmHgA (0-11.60 psi)
Carbon Dioxide Partial Pressure:	
- continuous exposure	< 1.5 mmHgA (.03 psi)
- short-term exposure	< 20.0 mmHgA (.39 psi)
Humidity Range	40-70% relative humidity
Temperature	
Desired Range	60-80 deg F
Metabolic Heat Production	100-1,000 Watts

In addition to the life support requirements, the spacesuit must also protect crewmembers from environmental hazards such as micrometeorites, space debris, radiation, and even sharp objects on other spacecraft and equipment (particularly due to the risk of puncture and pressure loss). In order to meet these many requirements, spacesuit designers have usually opted for a soft suit with many layers of different materials which perform various functions. In fact, the Apollo space suit had 21 layers according to the Du Pont company (Kozloski 1994). The Extravehicular Mobility Unit (EMU) used in the Space Shuttle program is also a fabric suit made up of many layers. The liquid cooling and ventilation garment (LCVG) of the EMU is made from nylon and spandex with a tricot lining. Cooling tubes made from Ethylene-vinyl-acetate are woven through the spandex. Next comes the pressure garment, which is made of urethane coated nylon covered by a woven dacron restraint layer. The outermost layers make up the thermal and micrometeoroid protection garment (TMG). The liner of the TMG is neoprene-coated ripstop nylon, followed by five layers of aluminized mylar thermal insulation, and finally a woven blend of kevlar and nomex synthetic fibers coated with teflon makes up the familiar white outer layer (Newman and Barratt 1995).

It is a great tribute to the spacesuit designers that a human is able to move at all when encumbered by this many layers. The most significant impact on astronaut mobility and workload in a fabric suit is increased pressure caused by a reduction in the volume of the suit when a joint is flexed. To minimize this effect in fabric suits, designers have incorporated axial restraint lines which hold the centerline of a joint at a near-constant

length. The volume decreases on the inside of the joint which is compensated for by expanding folds on the outside. Experimental hard suit designs like the NASA Ames AX-5 have almost entirely eradicated this effect by maintaining a constant volume during movements. The fact remains, however, that spacesuits still impose significant constraints on human movement. In addition to limiting joint range of motion, the mechanical properties of suits (i.e., elasticity, friction, and damping) and the mass properties of the associated suit segments impact EVA dynamics and can not be ignored. It is highly desirable to be able to simulate these various effects and predict their impact on specific EVA tasks. In addition, while a PLSS avoids the hindrance of umbilical lines, it must be realized that this massive backpack significantly displaces the center of mass of the combined astronaut and EMU. This can have surprising results in body translation or rotation, especially for partial gravity locomotion, and represents another important factor to be considered in dynamic simulation.

2.4 Training and Physical Simulators

Due to problems encountered in early EVAs, mission planners and training personnel began looking for ways in which they could better prepare astronauts for the unfamiliar environment encountered in EVA. One way of addressing the problem was to construct physical simulators. The main goal of these simulators was to try to give astronauts a feel for the way objects responded in weightlessness or in partial gravity. Some simulators sought to provide the additional degrees of freedom available to a body in space, while others aimed at imitating hypogravity conditions for locomotion studies (particularly in preparation for the Apollo moonwalks). During the past three decades a great variety of facilities have been built, many of them still in use today. They include neutral buoyancy tanks, air bearing floors, aircraft flying Keplerian trajectories, and various types of suspension mechanisms. Free fall towers and sleds were also used, although not often for human studies and even more seldom, if ever, for EVA studies. A review of the various techniques was recently published (Davis and Cavanagh 1993). A short description of the most important facilities used for simulating EVA, along with their advantages and disadvantages, is presented below.

2.4.1 Neutral Buoyancy Tanks

One of the early proponents of neutral buoyancy facilities was Werner Von Braun. At his behest, a large water tank was constructed at NASA's Marshall Space Flight Center. A picture of astronauts training for an EVA in this facility is shown in Figure 2.1. Since then, other tanks have been constructed, including the Weightless Environment Training

Facility (WET-F) at Johnson Space Center, a small tank at NASA Ames Research Center, and private facilities at McDonnel Douglas in Huntington Beach, CA and the University of Maryland. Weightlessness or partial gravity is simulated by adjusting ballast weights and air containers until gravity is canceled to the desired extent by the buoyancy force of the water. Neutral buoyancy is better suited to studies of slow motion tasks such as range of motion studies (Reinhardt 1989), but has also been used successfully for locomotion studies (Newman and Alexander 1991).

Advantages of this technique include the large working volume, an almost unlimited test duration, and the ability to approximate the full six degrees of freedom. A major disadvantage is the presence of hydrostatic drag and water displacement inertia. Astronauts have often expressed surprise at how much more easily objects move in the vacuum of space when unencumbered by the retarding forces present in the water tank. In fact some astronauts have even observed that their neuromuscular system sometimes appears to "learn" motion patterns during simulations in the neutral buoyancy facility that subsequently become a hindrance in the actual EVA performed in space. Furthermore, as every scuba diver knows, buoyancy usually varies with depth due to the difference in density between water, air, and other materials, making it difficult to maintain neutral buoyancy during depth changes.

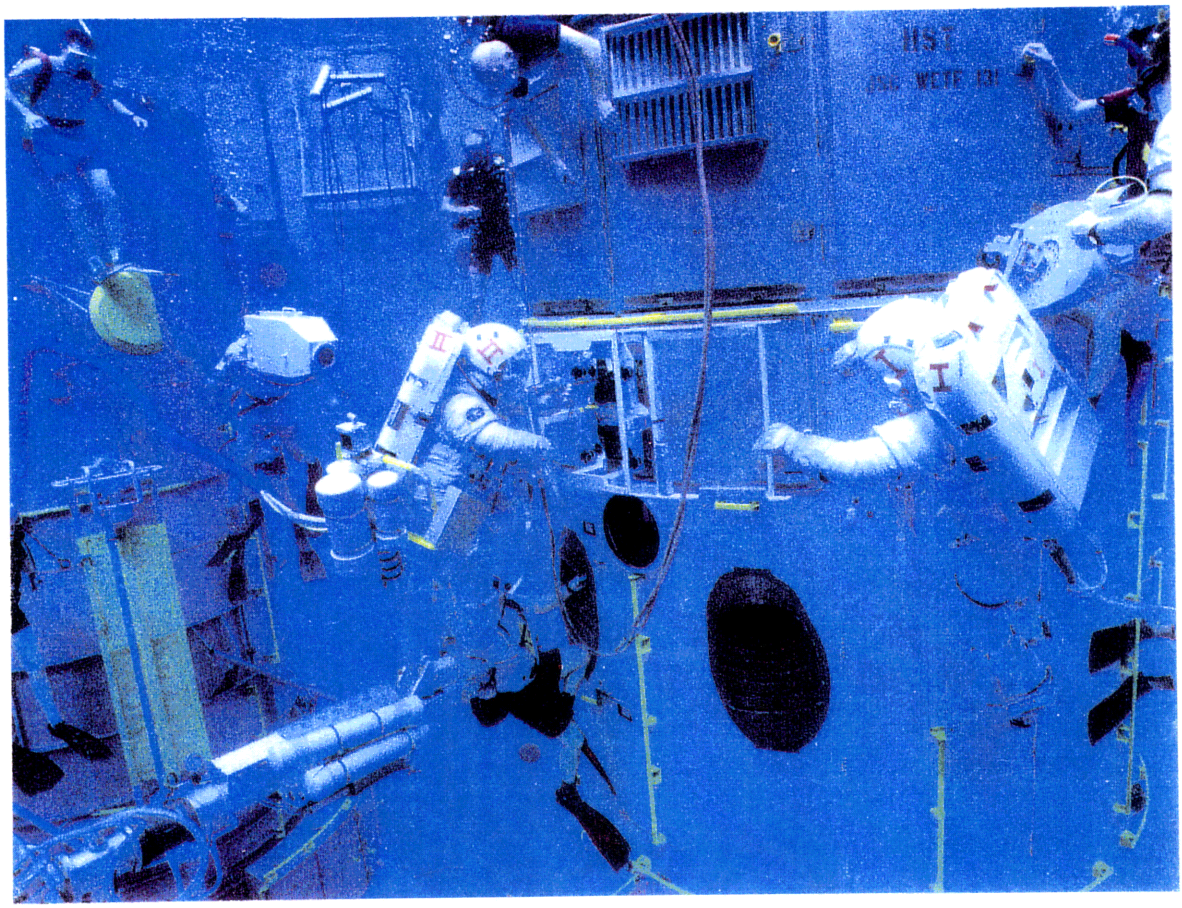


Figure 2.1 Astronauts during EVA training in the Neutral Buoyancy Facility at Marshall Space Flight Center. (Source: NASA)

2.4.2 Air Bearing Floors

Air bearing floors have proven useful mainly for translation tasks. The floor itself must be very flat to obtain the desired low friction. Usually stainless steel is used for this purpose. Each separate object to be simulated, be it a satellite or an astronaut, is supported by three or more flat pads in order to distribute the weight in a stable way. Air is expelled through the bottom of these pads and enough pressure is generated to support the entire object so that it can coast over the floor with very little friction. These air pressure pads are referred to as "air bearings". This technique of simulation has proven particularly useful for translation and rotation motions with devices such as the manned maneuvering unit (MMU). A picture of a spacesuited astronaut conducting a simulation on an air bearing floor is shown in Figure 2.2.

The main advantage of this facility is that friction and damping forces are very low. In addition, the simulation time is relatively unlimited and it is easier to set up test sessions on the air bearing floor than in facilities such as neutral buoyancy tanks and parabolic aircraft. An important disadvantage of the simulator is the limitation on the number of degrees of freedom. The air bearing floor itself, being a plane, can only accommodate three degrees of freedom, two in translation and one in rotation. Other degrees of freedom might be accommodated by a combination of mechanical bearings and/or suspension systems although increasing the mechanical complexity often introduces other problems such as static friction or inertia. Furthermore, the low friction level can be misleading. Friction in the air bearing operation, and drag from the surrounding air sometimes mask small forces and torques in the dynamic system which can turn out to be of critical importance in the frictionless vacuum of space. This effect was a significant factor in the difficulties encountered in the Intelsat VI capture EVA discussed later in this chapter.

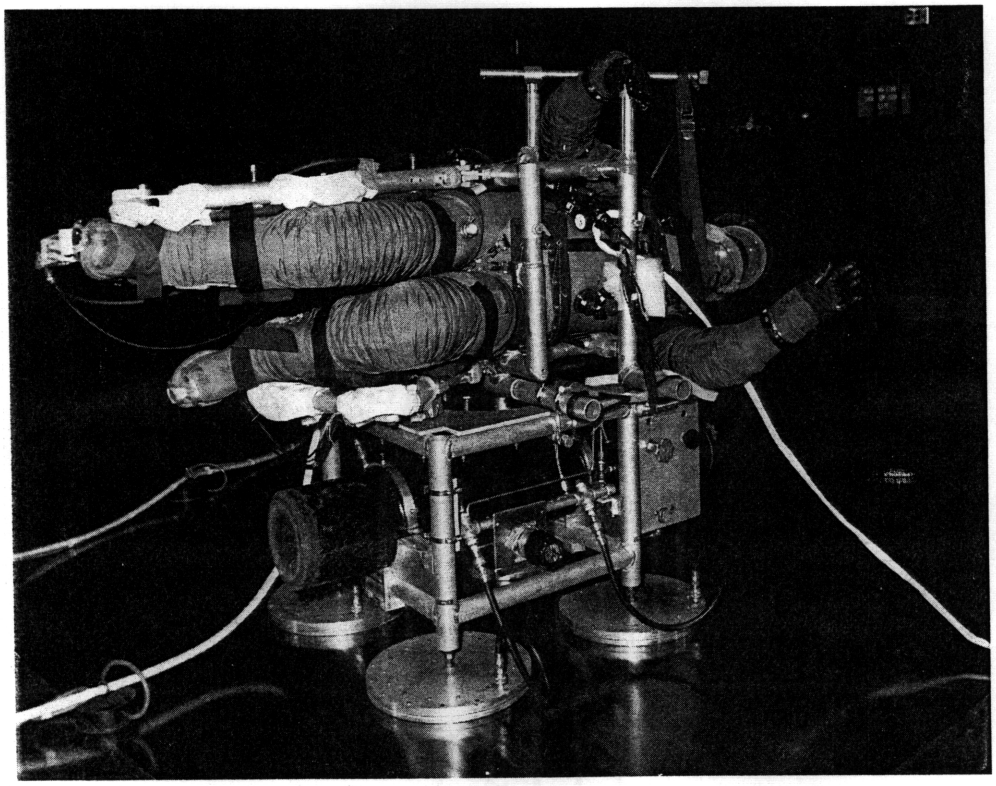


Figure 2.2 Spacesuited astronaut conducting a test on an air bearing floor (Kozloski 1994).

2.4.3 Aircraft Flying Parabolic Trajectories

NASA, ESA, the Russian Space Agency, and even some private companies such as Weaver Aerospace, operate aircraft which are capable of flying parabolic (or Keplerian) trajectories to achieve weightlessness through free fall. NASA's KC-135 aircraft (shown in Figure 2.3) was originally commissioned to enable new astronauts to gain experience in weightlessness prior to spaceflight. Since then, the KC-135 and other aircraft have been used for many scientific experiments. Occasionally these aircraft are also used for training in EVA tasks such as getting in and out of a foot restraint for instance.

The parabolic aircraft technique is popular because it offers a very realistic simulation of weightlessness. People and objects can readily achieve six degrees of freedom by floating freely within the cabin. It is also possible to simulate hypogravity or hypergravity by altering the aircraft trajectory as appropriate. Intravehicular tasks or experiments in which weightlessness plays a significant role are usually simulated using this method. The greatest disadvantage of parabolic aircraft is the limited duration of uninterrupted weightlessness. Only 25 seconds of "zero-g" are sustained on average per parabola. To make up for this, the pilots perform repeated parabolas. Unfortunately, the alternating g-level tends to provoke motion sickness in most passengers, hence the "vomit comet" nickname for the KC-135. Motion sickness is particularly dangerous for a spacesuited astronaut due to the risk of emetic clogging air lines or choking the astronaut. Another significant constraint is the limited volume (ranging from about 20 m^3 to about 100 m^3) and low vertical clearance (usually not more than 2.5 m) available in these aircraft. This is a serious limitation since many EVAs involve the manipulation of large objects. For these reasons, EVA simulations are most often performed in neutral buoyancy facilities while IVA (intravehicular activity) simulations are most often performed on parabolic aircraft.

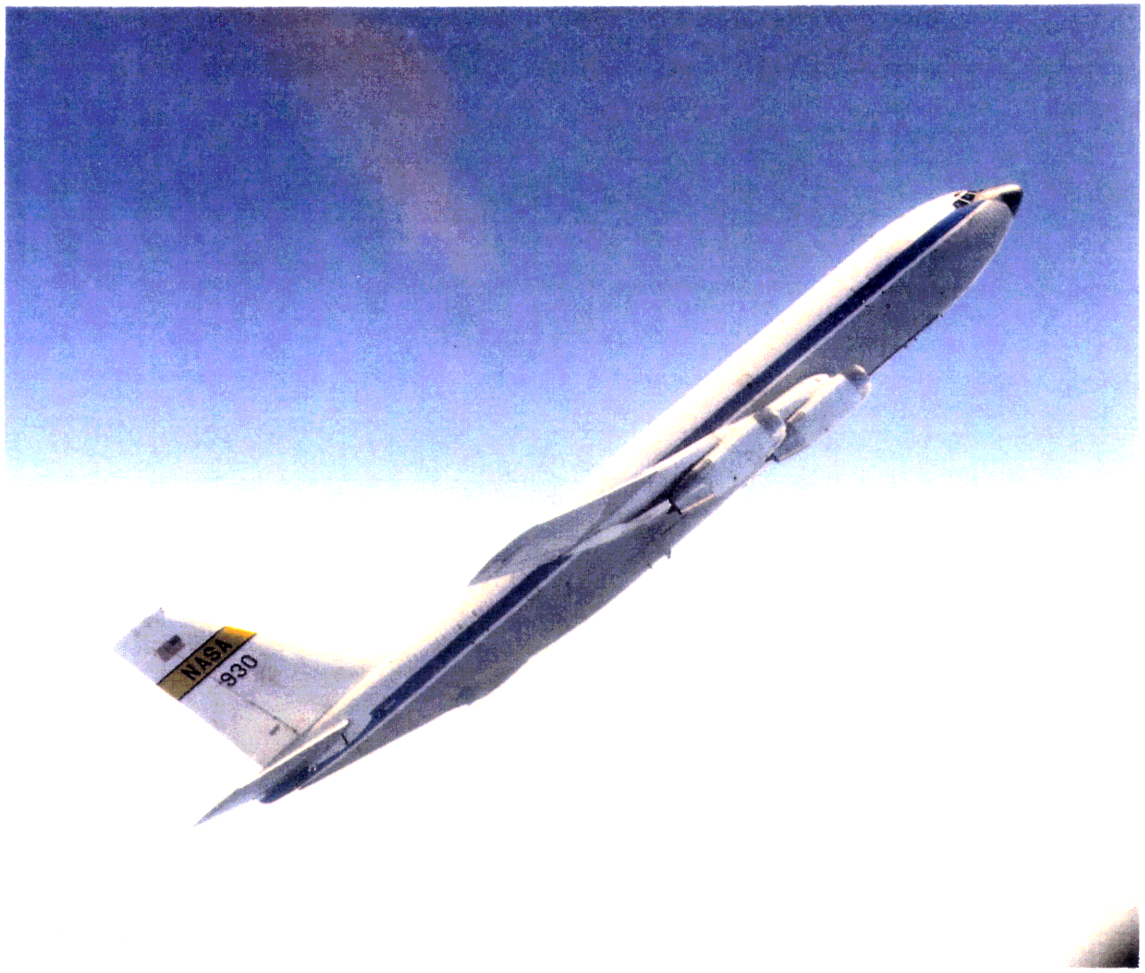


Figure 2.3 NASA KC-135 "Zero-g" aircraft entering a weightlessness parabola. (Source: NASA)

2.4.4 Suspension Mechanisms

Suspension mechanisms were quite popular during the early days of the Apollo program. Researchers used cables, pulleys and springs to investigate such topics as the energy costs of locomotion in reduced gravity (Wortz and Prescott 1966), or the mechanical aspects of vertical jumping in different gravity levels (Cavagna, Zamboni et al. 1972). Following Apollo the use of suspension systems diminished, but recently renewed interest in manned planetary exploration has prompted new studies (He, Kram et al. 1991). Most recently, the POGO pneumatic suspension system at Johnson Space Center (shown in Figure 2.4) was refurbished for EVA simulations including object manipulation for the Hubble Space Telescope repair mission.

Suspension mechanisms are useful because they allow a degree of freedom in the vertical direction, something which is hard to achieve using air bearing floors for instance. Also, reduced gravity levels can be simulated. By combining the suspension system with other mechanisms, such as tracks or gimbals, it is possible to approximate all six degrees of freedom. The major disadvantage of suspension systems is their limited range of effectiveness. It is particularly difficult to construct a suspension mechanism which can translate smoothly during forward motion. Also, many suspension systems make use of long springs which have an inherent limitation, the suspension force varies linearly with displacement. Although these limitations are averted in pneumatic suspension systems, like the POGO, they suffer from other detriments such as friction and pressure regulation.

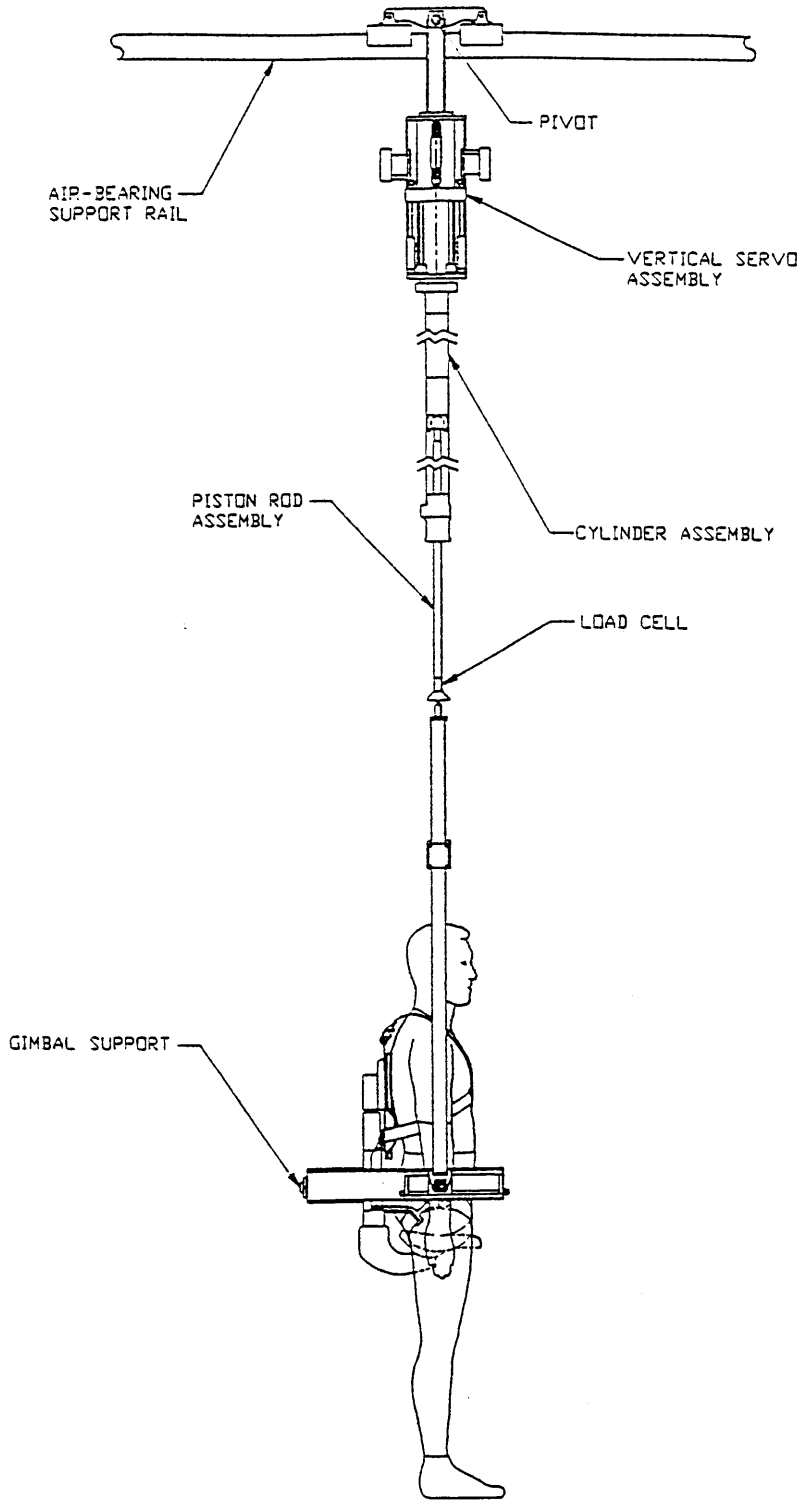


Figure 2.4 POGO suspension mechanism at NASA's Johnson Space Center. (Source: NASA)

2.4.5 Synopsis of Physical Simulators

All the physical simulators used to date have significant limitations in their ability to imitate weightlessness. While these simulators are very useful for research and training in certain types of EVA operations, they are constrained by physical realities. The use of a combination of simulators tends to compensate for the particular shortcomings of each, but it is still not possible to exactly duplicate the effects of weightlessness and vacuum, as experienced in orbit. Occasionally, physical simulators may fail to predict a certain effect or, through misrepresentation of the space environment, introduce habits which are detrimental during the performance of the actual EVA. An additional factor not mentioned above is that many of these simulators are expensive to operate. This is particularly true of the neutral buoyancy tanks and the parabolic aircraft.

As mentioned earlier, it is expected that hundreds of hours of EVA time will be required to assemble the International Space Station. Even if the current facilities were able to perfectly simulate weightlessness, they would not be able to accommodate the tremendous number of hours of training time required and, furthermore, tremendous costs would be incurred. Not only space station EVAs are of interest, planning for other types of EVAs, such as Hubble telescope servicing missions and possibly Lunar and Mars operations, must continue.

Clearly, to meet this growing need, it is desirable to find other ways of simulating EVA operations, particularly if these methods prove economical in terms of both finances and time. The primary goal of the research effort described in this thesis is to demonstrate how computational dynamic simulation can avoid the limitations of physical simulators conveniently and economically and at the same time yield numerical results hitherto unobtainable.

2.5 Computational Simulation of Multibody Dynamics

The discipline of multibody dynamic simulation has progressed to the point where it is now practical to simulate relatively complex dynamic systems fairly accurately, offering an additional way to simulate extravehicular activity with the advantage of avoiding many of the deficiencies of physical simulators. This method also has other attractive features such as low cost, short turnaround time, low manpower requirements, and ease of operation. Although this approach has its own limitations, such as mathematical complexity and model accuracy, its value lies in the fact that it can potentially make up for many of the gaps in the combined capability of physical simulators. For instance, a computational simulation would be capable of achieving the full six degrees of freedom that many physical simulators lack, and at the same time

avoid such detrimental effects as friction (air bearing floor) or viscosity (neutral buoyancy). While it is possible to simulate all objects comprising the dynamic system associated with an EVA, particular emphasis has been given to the goal of modeling the dynamics of the EVA crewmember's body as a system in itself.

At this point it is helpful to introduce some terminology used to describe the topology of multibody systems and this is done with the aid of Figure 2.5. An open Chain structure (Figure 2.5 (a)) is defined as a system of rigid bodies linked together in series. There are no loops. Only one of the bodies may be attached to ground since if two bodies were linked to ground it would constitute a loop structure. A loop structure (Figure 2.5 (b)), or closed chain, contains bodies that are linked together so that if one proceeds sequentially from one body to the next, one will end up back at the first body. Closed chains are difficult to model mathematically due to the additional constraints imposed and many multibody computer programs are incapable of analyzing them. A tree structure is usually defined by a base segment (or body) to which is attached two or more open or closed chains. A human-like example is shown in Figure 2.5 (c) and consists of a base segment (the trunk) to which are attached five open chain structures (the head, two arms, and two legs). The joints used to link the bodies in these structure may have degrees of freedom ranging from one to six (three translational and three rotational).

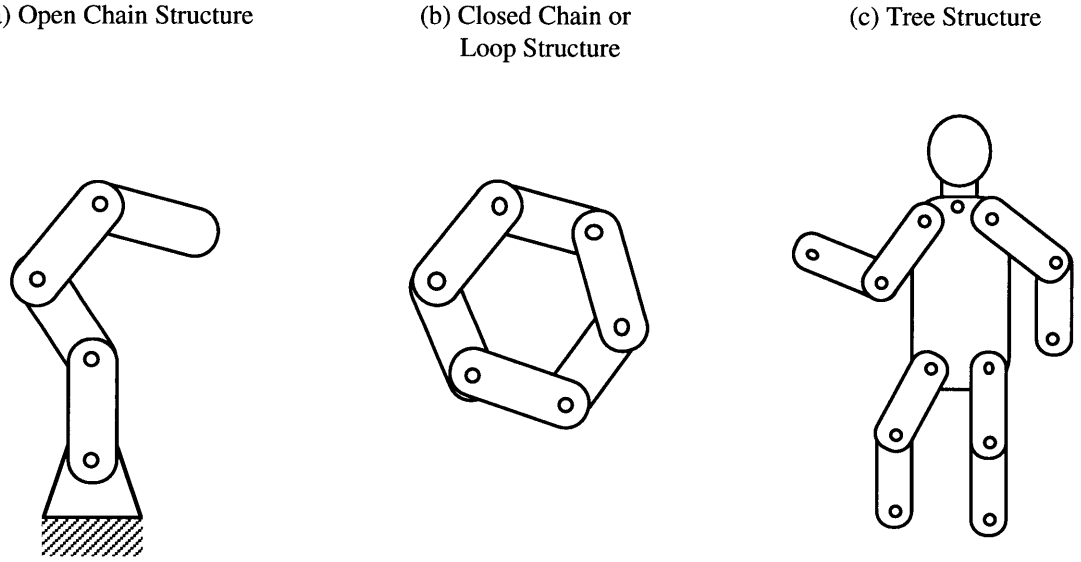


Figure 2.5 Topological terms applied to multibody dynamic systems.

The great scope of topics encompassed within the science of human movement is discussed in a CRC Critical Review in Biomechanical Engineering (Hemami 1985). Although slanted toward neuromuscular physiology rather than multibody dynamics, the

review points out that physiological systems are so complex that conventional mathematical methods such as solution of differential and integral equations of motion, analysis of stability, and systems of algebraic equations are incapable of producing significant results for all but the simplest systems. Instead, researchers have been forced to resort to other methods, particularly those involving use of the digital computer. The review examines human movement in terms of three areas of concentration: (1) mathematical modeling of the human body, (2) control methods, and (3) simulation methods and displays. In addition to providing an overview of the work in the field of human movement science, including a very extensive list of references, this report also identifies questions for further research such as issues of control, neural communication, coupled and decoupled oscillators, and cerebellar models.

Computational simulation of multibody dynamics refers to the mathematical modeling of the motion of an interacting system of bodies driven by forces and torques. The physical principle upon which all of these simulations are fundamentally based is Newton's second law (the force on a particle is equivalent to its mass multiplied by its acceleration). Rigid body dynamics is essentially an extension of this law to a set of distributed particles, hence introducing the inertia tensor. Multibody dynamics is the extension of Newton's principle to a system of rigid bodies. Flexible body dynamics have also been studied, representing a far more complex situation mathematically. Often, a flexible body is modeled simply as a chain linkage of rigid bodies. The human body, like a spacecraft or robot, can be modeled as a multibody system. In reality, the mechanics of human anatomy are fairly complex. Joints are made up of a combination of passive compressive components, cartilage; and passive extensive components, ligaments; and the geometry of interaction between segments can be quite involved. Nevertheless, it has been found that a high degree of accuracy and meaningful results can be obtained by modeling human body segments as rigid bodies connected by simple joints, such as pin joints or ball joints, according to the required number of degrees of freedom in the model (Hemami 1985).

Recently, an effort has been made to apply the simulation capabilities of computers to EVA (Price, Fruhwirth et al. 1994). The work done at McDonnel Douglas Aerospace by Price et al. uses a Computer Aided Design (CAD) approach to obtain three dimensional rendered images of astronauts and objects involved in EVA tasks. This type of program is useful for visualization, fit checks, and anthropometrics, but does not include any dynamic simulation capability.

Multibody dynamics are discussed in more detail below. The first subsection gives an overview of the evolution of the most common classical and contemporary dynamical

equation formulation methods, the computer programs developed to execute simulations, and user interfaces such as animation schemes. Subsection 2.5.2 focuses on one particular method, Kane's dynamical equations. This formulation, named after Professor Emeritus Thomas R. Kane of Stanford University, is the basis for two multibody dynamics programs, namely, AUTOLEV (Schaechter and Levinson 1988) and SD/FAST (Rosenthal and Sherman 1986). Kane's method and SD/FAST are described in more detail because these are currently the tools being used at MIT to conduct the EVA dynamics research presented in this thesis. The last subsection, 2.5.4, discusses some of the most significant limitations of computational simulation of multibody dynamics and identifies some of the challenging issues.

2.5.1 Formulation of Dynamical Equations and Simulation Methods

Several formulations for determining the equations of motion of a multibody dynamic system have been developed. The most important methods are the Newton-Euler, Lagrange, D'Alembert, Hamilton, Boltzmann-Hamel, Gibbs-Appell, and Armstrong recursive algorithm. Rather than dealing with each of these methods in detail, which would fill a sizable textbook, this section presents short descriptions of the most relevant publications in an attempt to give a synopsis of the evolution of multibody dynamics formulation and simulation, the range of methods employed, and their strengths and weaknesses.

The first papers describing general multibody dynamics programs appeared in the mid 60's (Hooker and Margulies 1965) (Roberson and Wittenburg 1966). Most of these early efforts were based on either the Newton-Euler equations, Lagrange's equations, or a combination and were usually applied to the case of spacecraft dynamics. Another early attempt at analyzing the dynamics of a complex multibody system was performed, interestingly enough, on the subject of astronaut motion in free fall (or weightlessness) (Scher and Kane 1969). The goal was to describe how an astronaut might be able to change the orientation of his body in space while obeying the law of conservation of angular momentum. The equations were derived by hand and, in the words of Prof. Kane, it took "days simply to find missed minus signs." Researchers studying multibody systems of similar complexity, be it robots or vehicles or mechanisms, were beginning to realize that a computer program capable of formulating the dynamical equations of these types of linked systems and of carrying out motion simulations would be a very powerful and time saving tool.

Advances in the capabilities and performance of computers have greatly increased the amount of research in the area of multibody dynamics, including human body dynamics,

over the last two decades. An early example is the analysis of human body dynamics in vehicle collisions (Wittenburg 1979). The body was modeled as eleven linked rigid bodies with a total of 27 degrees of freedom. The dynamical equations were derived from D'Alembert's principle. Due to the complexity of the nonlinear differential equations it was desirable to perform the formulation only once and in such a way that the equations would be applicable to any system that formed a tree structure, regardless of the type of joints and the number of segments. Matrix methods were used to accomplish these goals. Generalized coordinates, obtained as functions of time from photographic recordings, combined with measurements of external forces, were used as input data. The unknown muscle forces could then be determined from time-variant linear equations.

A similar human motion study was performed by (Ramey and Yang 1980), this time with the intention of simulating motion associated with sport, dance, and other vigorous activities. The equations of motion were determined from the principle of conservation of angular momentum, defined with respect to the center of mass of the nine segment body model, and described in detail for the case of torque free motion. Coordinate transformations were also illustrated. The main objective was to extend simulation methods to the more general case of human motion in three-dimensional space. To allow more degrees of freedom (d.o.f.), the upper arms and thighs were attached to the trunk by ball-and-socket joints (3 d.o.f.), while the elbows and knees were modeled by simple hinge joints (1 d.o.f.). A FORTRAN program was used to perform the integration of the equations of motion during simulation runs. The simulation was conducted under conditions of free fall and applied to a hitch-kick and a somersault type long jump. As with the Wittenburg study, the motion time histories were estimated by reviewing films of the tasks. With the guidance of a professional sports coach the motion data was perturbed slightly and the simulations rerun to determine the change in muscle forces resulting from the revised movement.

The complexity of multibody dynamics places severe demands on the performance of computer simulations. In the early 80's, there was a lot of interest in improving the efficiency of computer simulation codes. A comparison of the computational complexity and execution times of four different methods of simulating a robotic mechanism was an example (Walker and Orin 1982). The first three methods utilized variations on the Newton-Euler formulation while the fourth was based on a recursive algorithm for computing the moment of inertia matrix. All four methods were applicable to open loop kinematic chains and were programmed in FORTRAN. Predetermined joint position, velocity, and force and torque values were used to obtain the joint accelerations. It was observed that the Newton-Euler schemes were more efficient, the computation time

generally increasing linearly as degrees of freedom were added. Occasionally the recursive scheme took less time to execute due to early convergence of the iterations. However, the third Newton-Euler scheme was the most efficient since it took advantage of the symmetry of the inertia matrix and used a recursive scheme to determine the mass properties.

It is generally believed that the classic Lagrangian formulation is less efficient than most other schemes due to the many steps required to derive and differentiate the kinetic energy equations. It has been shown, however, that the efficiency depends upon the form of representation of the rotational dynamics and the recursive structure of the computations. In fact, with the appropriate choice of these descriptions, the Lagrangian formulation is equivalent to the Newton-Euler formulation and there is no fundamental difference in their computational efficiencies (Silver 1982).

Not all researchers confined their efforts to classical methods of multibody dynamical equation formulation. Given the actuator forces and torques, Featherstone tried an alternative approach to calculating the acceleration of robot segments (Featherstone 1983). This method was based on a recursive algorithm utilizing values called "articulated-body inertias" that represented the overall inertia properties of a system of rigid bodies connected by joints that allowed constrained relative motion. These values, and other spatial quantities, were represented by a matrix-based notation. The results indicated that while the algorithm was convenient to employ and the computational requirement varied only linearly with the number of joints, there were two major drawbacks. Firstly, in a realistic sense it was applicable only to open-loop kinematic chains with revolute and prismatic joints. Secondly, it was less efficient than other composite rigid-body methods, except for very complex systems.

Occasionally, researchers were able to make improvements in the efficiency and utility of conventional schemes. For instance, Marshall produced a simulation method based on the general Newtonian equations for an n-segment open chain model (Marshall, Jensen et al. 1984). The inputs to the program were the geometry, masses, and moments of inertia of the bodies; the time histories of joint moments and force; the initial absolute angular positions and velocities; and the acceleration time history for the constrained axis of the n^{th} segment. The output of the program were the angular accelerations, obtained by solving the n simultaneous linear equations, and the angular velocities and positions, obtained by performing forward integrations. Results were visualized by means of a computer graphics display of a linked figure with segments represented by straight lines of appropriate length. The simulation program was applied to a three segment model of the lower extremity during various movements and to a five segment model executing a

vertical jump. The study also examined the potential for altering input data, such as torques and inertial parameters, and discussed the implications of anticipating the effects of these changes.

A combination of techniques was used by Armstrong who formulated the equations of motion by means of the classical Lagrangian and Newton-Euler methods, but ran the simulations by means of recursive solutions of the equations of motion (Armstrong and Green 1985). This method, applicable to tree structures only, was a generalization of earlier work done by Armstrong (1979) on linear linkages, this time with improved computational efficiency. The primary goal was to use dynamics to create realistic animation. The animator, it was argued, should be able to manipulate the motion through the alteration of dynamics parameters as conveniently as one would use curves and surfaces to fit spatial data points.

Advances in the ability to formulate dynamical equations conveniently and drive simulations allowed researchers to consider other issues such as kinematic constraints and control methods. For example, Isaacs worked on the control of dynamic simulations with kinematic constraints, behavior functions and inverse dynamics (Isaacs and Cohen 1987). A program called DYNAMO (for DYNAmic MOtion system) was developed in which the equations of motion were derived using D'Alembert's principle of virtual work. The simultaneous equations of motion were represented in matrix form and solved using a simple Gaussian elimination scheme which made it possible to exert the types of control central to the DYNAMO scheme. It was pointed out that while others had used more efficient recursive schemes, such methods did not allow the combination of prescribed force and prescribed motion in the same simulation. Three means of achieving control were introduced: (1) *kinematic constraints* which allow traditional keyframe-type animation and incorporate joint limits; (2) *behavior functions* which relate the instantaneous state of the system to the desired accelerations and forces; and (3) *inverse dynamics* in which the joint forces and torques required to achieve the desired kinematics are determined. A year later, Isaacs presented an expansion of their DYNAMO program with enhanced user control of dynamic simulation (Isaacs and Cohen 1988). Forward and inverse kinematics were combined as a means of prescribing motion in a mixed forward and inverse dynamics simulation. Kinematical constraints were incorporated into the mathematical formulation of the dynamical equations by means of a Lagrange multiplier method, thus enabling the simulation of keyframed paths, closed loops, point-to-path constraints and the interaction between links and the environment. The unknown motion and constraint forces were determined by a simultaneous solution method. Three example

simulations were presented: a chain forming a closed loop, a chain interacting with a floor, and a marionette.

Control issues were also addressed by researchers who studied dynamic simulation within the context of computer graphic simulation (Wilhelms, Moore et al. 1988). The motivation for using dynamic simulation in the Wilhelms study was the promise that complicated motions could be obtained with less user interaction, although at the expense of greater computational requirements. Two types of issues were examined: low level control issues such as collision response, elasticity, friction, joint limits, damping, and motion constraints; and user interface issues such as interface design, menu options, and integration within an overall animation program. The name given to the animation system was "Kaya". Low level control was defined in terms of basic Newtonian physics and the control of segments as rigid bodies. To form the dynamical equations of motion and run simulations, a recursive Armstrong (1985) formulation based on the Euler formulation and numerical integration methods was used. Pre-processing control was performed along with matrix control in which the constraints became part of the matrix.

An important issue in human body simulations is the accuracy with which human body models represent the actual kinematics and dynamics of the human body. One particular concern is the nature of the joints used, degrees of freedom, and accuracy. A study on this topic was conducted by Kinzel on kinematic models of anatomical joints with from one to six degrees of freedom (Kinzel and Gutkowski 1983). Five joint types were examined in detail: the one-d.o.f. revolute joint, the three-d.o.f. planar joint, the three-d.o.f. spherical or ball and socket joint, the two-d.o.f. spherical joint, and the six-d.o.f. spatial joint. A discussion of available motion measurement and description techniques was also included as well as an extensive review of literature on anatomical joint kinematics.

Also, Calderale conducted a review of the mathematical models of musculoskeletal systems, examining the methods used to evaluate muscle forces and joint forces and torques (Calderale and Scelfo 1987). The studies discussed were based on quasi-static rather than dynamic principles. Theoretical models were classified as statically determinate or statically indeterminate. Experimental models were characterized by the estimation of muscle forces from EMG data. The relative simplicity of a non-dynamic approach allowed the researchers to account for greater detail in the geometry of the joints and segments. Specific topics covered included the forces produced by muscle groups, internal joint reaction forces, and isometric and isokinetic contraction.

2.5.2 Kane's Dynamical Equations, AUTOLEV and SD/FAST

Kane has been one of the pioneers of multibody dynamics. Some of his early work addressed the dynamics of the human body in weightlessness, a topic closely related to the research described in this report. Kane was concerned with the difficulties experienced in formulating equations of motion and the efficiency of computer simulations. His efforts, with the help of his students and colleagues, have led to great advances in the efficiency and utility of multibody dynamic simulation, so much so that his unique formulation method now bears his name.

About a decade ago, Kane published a paper in which he suggested ways of addressing the most important issues in the field of multibody dynamics at the time (Kane and Levinson 1983). He discusses several difficulties associated with the use of the Newton-Euler equations and Lagrangian equations. In the case of the Newton-Euler formulation it is the great effort required to eliminate nonworking constraint forces. In the Lagrangian formulation, on the other hand, it is very time consuming to derive and differentiate the kinetic energy expressions by hand and algebraic errors are hard to avoid. In addition, the resulting equations are difficult to manipulate and the significance of individual terms is not clear. The major issue, Kane argues, is to determine which method provides the best means of solving multibody dynamics problems. While the difficulties encountered with the Newton-Euler constraint forces are readily observed in simple systems, the difficulties associated with the Lagrangian formulation and virtual work methods only become apparent in more complicated systems. Kane compares these two methods with four variations of his own formulation method for the dynamical equations. He identifies a fundamental flaw of the Lagrangian formulation as the inability to deal directly with dependent variables other than the generalized coordinates. He states that formulations obtained through Kane's dynamical equations proved to be as much as four times faster than conventional multibody programs in terms of CPU time. This improvement stems from the fact that an explicit computational algorithm is formulated based on Kane's equations, which is specific to the system under analysis. This contrasts with conventional programs where equations are formulated in their most general form and use extraneous operations during simulation.

Kane and Levinson extended the application of Kane's dynamical equations to the field of robotics (Kane and Levinson 1983). Two application modes are identified: *simulations*, which are used to test performance in a theoretical sense, and *operations*, in which the dynamical equations are used to compute the necessary forces and torques produced between the segments for a desired end effector motion or force. In the latter case, repeatability and speed are important. They point out that most general purpose

multibody dynamics computer programs created for the numerical formulation and solution of equations of motion are slow and inefficient. This is largely because they use the Lagrange or Newton-Euler methods which are subject to the limitations mentioned above. The advantages of Kane's dynamical equations, they explain, is that they allow one to work with dependent variables tailored to the specific problem, to avoid calculating unimportant forces and torques, and to produce computationally efficient equations of motion in an explicit form. The basic steps in the formulation of Kane's Dynamical equations involve obtaining generalized speeds, partial angular velocities, partial velocities, generalized inertia forces, and generalized active forces. For complete mathematical detail on how these steps are carried out, the reader is referred to the texts *Spacecraft Dynamics* (Kane, Likins et al. 1983) and *DYNAMICS: Theory and Applications* (Kane and Levinson 1985).

Kane's students and colleagues have used the advantages inherent in his formulation to create the two well known and powerful multibody dynamics analysis programs AUTOLEV (Schaechter and Levinson 1988) and SD/FAST (Rosenthal and Sherman 1986) mentioned previously. Schaechter and Levinson describe AUTOLEV as computerized symbol manipulation for the formulation of equations of motion tailored to a specific system. AUTOLEV generates FORTRAN code to use within a simulation. The technique is based on Kane's method and is designed to imitate the process a human analyst would follow. Examples of multibody spacecraft systems are used to illustrate the use of the software. Rosenthal and Sherman describe a program called SD/EXACT, which served as the foundation for SD/FAST. Since SD/FAST is the program used for the analysis described in this thesis, it is worth discussing its features in a little greater detail.

Rosenthal and Sherman believe that there are two important factors in the run time performance of a multibody dynamics program: the inherent complexity of the multibody formalism used as the basis for simulation, and the efficiency with which that formalism is converted to computer code. To illustrate the former factor they contrast two ways of deriving the equations of motion for a single rigid body: the Lagrange method and angular momentum principles. The Lagrange method is encumbered by the necessity of using generalized coordinates as dependent variables. This leads to equations which are long, contain many trigonometric functions, and are strongly coupled in the highest derivatives. On the other hand, the use of angular momentum principles, where angular velocity values are used as dependent variables, leads directly to Euler's dynamical equations which are compact, free of trigonometric functions, and uncoupled in the highest derivatives.

A similar situation occurs in regard to multibody systems. Kane's method leads directly to the simplest possible equations of motion. The classical methods (Newton-Euler, Lagrange, D'Alembert, Hamilton, Boltzmann-Hamel, Gibbs) do not. Even so, Kane's method does not guarantee that the algorithm will be in its most computationally efficient form. For instance, collections of terms which occur repeatedly should each be assigned to a distinct variable to avoid repetition. In order to do this, the equations of motion must be in an explicit form, which is not the case with matrix methods or other conventional multibody programs cited by Rosenthal and Sherman (MBDY (Hooker 1974), NBOD2 (Frisch 1974), TREETOPS (Singh and Vandervoort 1985), and DISCOS (Frisch 1975)). Furthermore, a practical multibody program must be able to support a wide variety of system topologies, which is a further challenge in creating efficient code, since the program must be able to represent the equations in their most general form. A large number of terms in the general equations of motion may be eliminated since real systems usually have bodies that have one or more axes of symmetry, thus producing zero terms in the mass matrix, and they are often connected by joints with perpendicular axes.

Rosenthal and Sherman developed a program, called SD/EXACT in their original publication but now known as SD/FAST, which achieves a very high level of efficiency by avoiding redundant operations. The efficiency is obtained by utilizing computer symbol manipulation. A general purpose multibody program creates specialized code to represent a particular multibody system. Computer symbol manipulation is then used to convert the equations of motion from their general form into their simplest form and FORTRAN subroutines or C functions are generated to represent the dynamics. According to Rosenthal and Sherman, a symbolic manipulation program follows a process very similar to that of a human analyst. The computer generated code is actually slightly faster than the most efficient code developed by a human analyst due to additional enhancements achieved through clever programming techniques. Both the human analyst and computer symbol manipulation methods are an order of magnitude faster than conventional multibody programs. The symbolic manipulation program, however, saves the analyst many hours (even days) otherwise wasted in formulating the equations of motion and also avoids the frustrating task of searching for the inevitable algebraic errors that occur in hand-formulation.

To show why SD/FAST is faster than other methods, Rosenthal and Sherman contrast the classical methods, represented by the Hooker-Margulies equations (Hooker 1970), and Kane's method. Both methods lead to equations of motion which can be written most compactly as Newton's famous second law:

$$M\dot{u} = F \tag{2.1}$$

where,

$M = n \times n$ positive definite "mass" matrix
$F = 1 \times n$ force vector
$u =$ Kane's generalized speeds

Calculation of the mass matrix accounts for the largest portion of computation time (since there are n^2 entries in the matrix as opposed to only n entries in each of the vectors). The typical approach taken in general multibody programs is presented first (Fleischer and Likins 1974). In Fleischer's method, the elements of the mass matrix are obtained from

$$m_{rs} = \mathbf{g}^r \bullet \sum_{k=1}^{B} \sum_{j=1}^{B} \varepsilon_{rk} \varepsilon_{sj} \Phi^{kj} \bullet \mathbf{g}^s \qquad (r,s = 1,....,n) \tag{2.2}$$

where,

g's = hinge vectors
ε's = "path elements"
Φ = "augmented inertia dyadic"

The most significant point is that each element is computed as a double summation over B, the number of bodies. Thus, determining the mass matrix requires $O(n^4)$ operations. By contrast, Kane's method utilizes a different formula for obtaining the mass matrix elements:

$$m_{rs} = \sum_{k=1}^{B} \left(\mathrm{m_k} \mathbf{v}_r^{k*} \bullet \mathbf{v}_s^{k*} + \mathbf{w}_r^k \bullet \mathbf{I}^k \bullet \mathbf{w}_s^k \right) \qquad (r,s = 1,....,n) \tag{2.3}$$

where,

m_k = mass of body
I^k = central inertia dyadic of body k
$\mathbf{v}_r^{k*}$ = Kane's partial linear velocity for body k
$\mathbf{w}_r^{k*}$ = Kane's partial angular velocity for body k

The algorithm requires only $O(n^3)$ operations due to the single summation. Thus, the more complex the system (the greater the number of bodies and degrees of freedom) the greater the improvement in performance of Kane's method over the classical methods. Equation (2.3) represents the core of the SD/FAST program.

In addition to this highly efficient algorithm, SD/FAST also has a built in symbol manipulator. It performs the appropriate simplifications as expressions are assembled. Equations of motion specific to a particular system and simulation are produced. The program also takes advantage of simplifications made possible by symmetry in the system. Finally, additional optimizations are achieved by the employment of efficient programming methods. For instance, terms which appear repeatedly are assigned to variables at the beginning and there are no loops in the main body of the generated code.

Due to the advantages of improved performance, ease of use, and versatility, SD/FAST (based on Kane's method) was used for the EVA simulations described in this thesis. It should be stressed, though, that while SD/FAST makes it possible to avoid much of the tedious work associated with formulation of the equations of motion for a multibody system, the researcher is still left with the significant task of creating the simulation code. Fortunately, this allows the analyst much flexibility in creating simulations and facilitates the modification of systems being studied.

Since it is difficult to interpret long columns of raw data when dealing with complex multibody systems, it is helpful to create programs which post-process the simulation data and provide the researcher with easily interpretable plots, preferably coordinated with some sort of animation. Much effort was devoted toward the creation of an interactive plotting and animation program. The creation of simulation code and a graphical representation program is discussed in detail in the next chapter.

2.5.3 Intelsat VI Capture Mission Simulation

Background on Intelsat Capture

During the Space Shuttle mission STS-49 in May 1992, astronauts performed an EVA to capture the Intelsat VI satellite stranded in a too low earth orbit. The original plan called for one astronaut to clamp a specially designed capture mechanism on to the structural interface ring in the base of the cylindrical satellite. Figure 2.6 shows the astronaut, capture bar, and satellite.

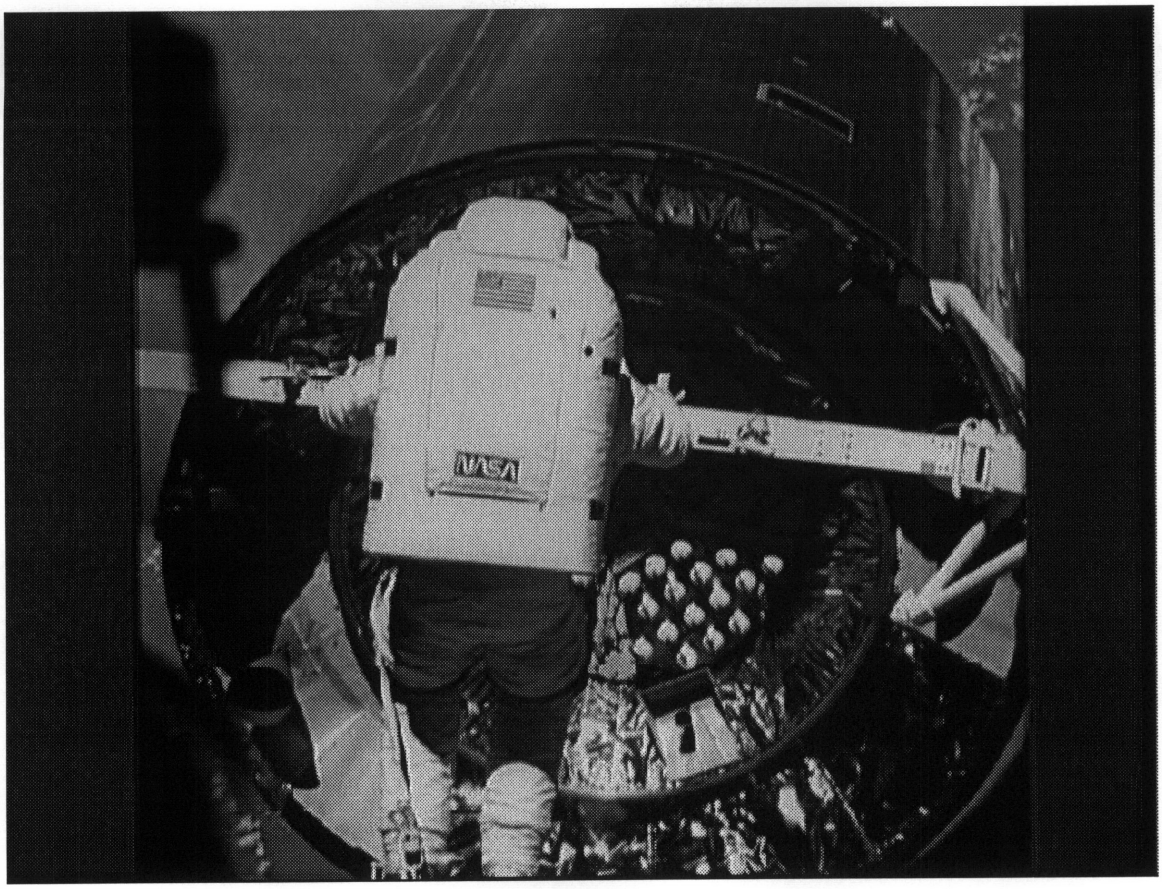


Figure 2.6 Astronaut attempting to attach capture bar to satellite interface ring. (Source: NASA)

Since the Intelsat VI did not have interfaces such as handholds to facilitate interaction by astronauts, the capture mission was quite challenging. In several attempts during two separate EVA sorties, the astronaut was unable to capture the satellite using the rehearsed procedure with the capture bar. Latches in the capture mechanism failed to fire before the slowly rotating satellite (about 1 rpm (revolution per minute)) started nutating and translated beyond reach. The capture bar had to be abandoned. Overnight, astronauts on board the Orbiter, together with mission control in Houston, devised an ingenious emergency EVA. Three astronauts secured at various points in the payload bay would grab the base of the satellite after the commander had flown the orbiter within arms reach. This procedure succeeded.

Although the mission was ultimately successful, there was considerable concern that the original planned procedure failed. A NASA review panel investigated the failed capture attempts and among the most important recommendations was the use of a six-

d.o.f. dynamic analysis. The following two sections describe an example of how such an analysis could be performed.

Intelsat Capture Simulation Method

A preliminary study adopting the philosophy of simulating the simplest possible dynamic system that would produce valuable results, was undertaken. Thus the analysis focused on the dynamics of the Intelsat VI satellite itself, while the interaction from the crewmember and capture bar were simply modeled as forces applied at the contact points between the capture bar and the satellite structural interface ring. The physical properties used to model the Intelsat VI satellite are given in Table 2.2.

Table 2.2 Physical properties of Intelsat VI satellite. (Holloway 1992; NASA 1992)

Property	**Value**
length	4.37 m
diameter	3.64 m
thrust-ring diameter	2.35 m
thrust-ring inset	.31 m
mass	4064.6 kg
C.G.	1.34 m forward of aft-end
I_{xx}	6140 kg-m^2
I_{yy}	6781 kg-m^2
I_{zz}	6114 kg-m^2
roll-rate (approximate)	1 rpm (6.28 radians/sec)

The Intelsat VI satellite was modeled simply as a rigid body with six degrees of freedom in space. SD/FAST was used to formulate the equations of motion of the dynamic system. The simulation was driven by specialized code developed at MIT. In establishing the conditions for the simulations, the philosophy was to try a number of scenarios, based on physical reasoning and discussions with EVA crewmembers, and then observe the resulting effects. A similar procedure would be followed if the simulations were performed prior to a mission. Three cases representing the range of scenarios considered are given in Table 2.3. The location and direction of the interacting forces and moments are shown in Figure 2.7. Note the convention used to label the forces: F_{ly} is the force applied to the left side, in the Y direction; F_{ry} is the force applied

to the right side, in the Y direction; F_{lz} is the force applied to the left side, in the Z direction; and F_{rz} is the force applied to the right side, in the Z direction. The convention used to label the moments is: My is the moment about the Y axis (roll); M_z is the moment about the Z axis (yaw).

Table 2.3 Three cases of forces and moments produced by interaction between the crewmember with the capture bar and the satellite.

Case	Description	Time	Forces (N)				Mom. (N-m)	
		sec	F_{ly}	F_{ry}	F_{lz}	F_{rz}	M_y	M_z
A	Single-sided contact by capture bar	1.0	0	44.48	0	0	0	52.27
B	Unbalanced two-sided contact by capture bar	3.0	31.14	53.38	0	0	0	26.13
C	Unbalanced two-sided contact with frictional counter-torque (initial roll-rate is nearly canceled after 3 sec)	3.0	31.14	53.38	-80.43	80.43	-189.01	26.13

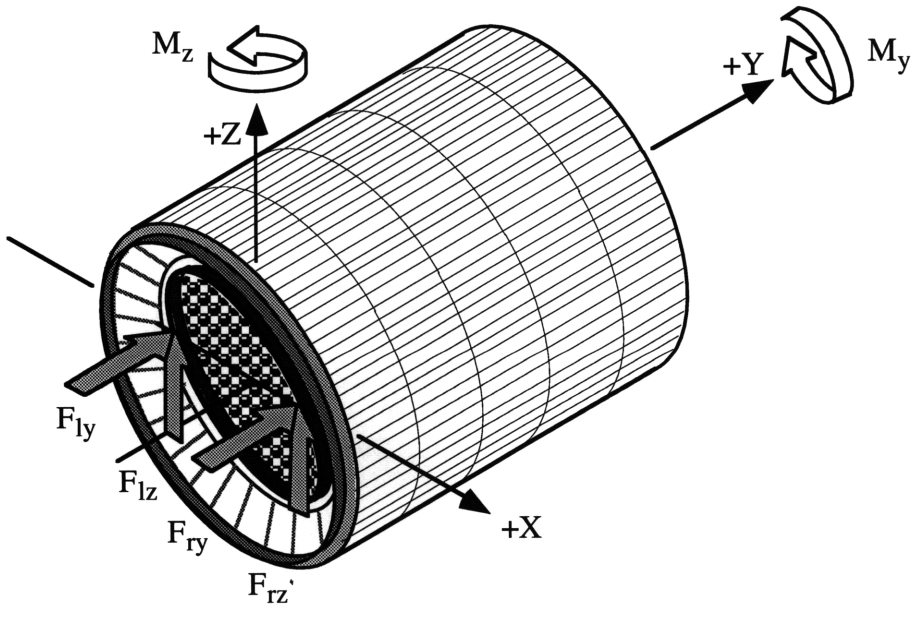


Figure 2.7 Forces and moments produced by the contact points of the capture bar on the satellite.

The response of the satellite to each of these cases was determined independently. Since the satellite is treated as a rigid body, its translational motion is simply governed by Newton's second law,

$$F = ma \tag{2.4}$$

where,

F = external force vector acting on satellite
m = mass of satellite
a = acceleration vector of satellite center of mass

while its rotational motion is governed by Euler's equivalent formulation,

$$\tau = I\alpha \tag{2.5}$$

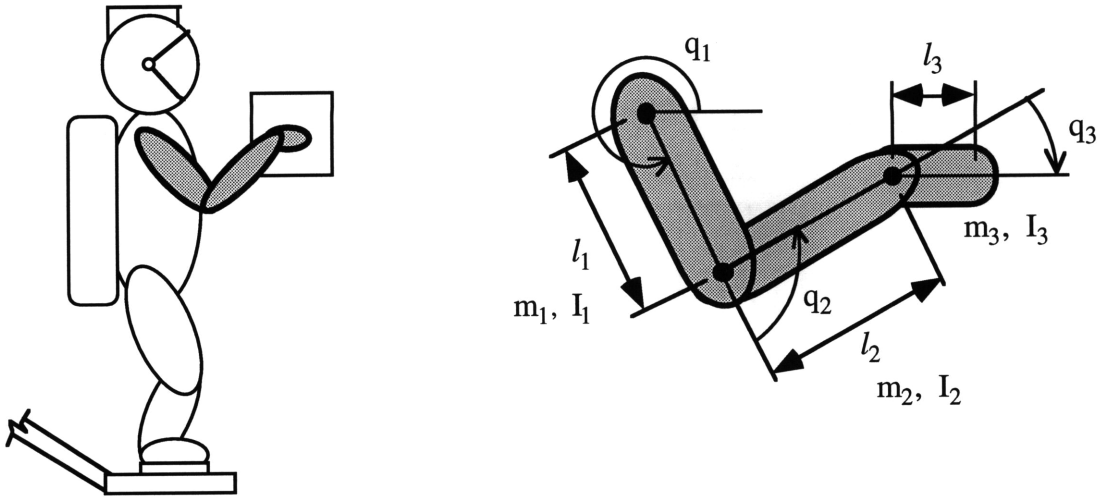
where,

τ = external torque vector acting on satellite
I = inertia tensor
α = acceleration vector

The motions of the crewmember's arms and the capture bar were simulated using inverse kinematics. The relationship between the velocity vector describing the capture bar motion, $\dot{p}$, and the joint velocity vector, $\dot{q}$ (describing the rate of change of the joint angles in the multi-link model of the crewperson's body), is given by

$$\dot{p} = J\dot{q} \tag{2.6}$$

where **J** is the Jacobian matrix relating the velocity of the capture bar, in endpoint coordinates, to the velocity of the astronaut's arms, in joint coordinates. The geometry is shown in Figure 2.8.



Definitions:

q_1 = shoulder joint angle	q_2 = elbow joint angle	q_3 = wrist joint angle
l_1 = upper arm length	l_2 = forearm length	l_3 = hand length
m_1 = upper arm mass	m_2 = forearm mass	m_3 = hand mass
I_1 = upper arm moment of inertia	I_2 = forearm moment of inertia	I_3 = hand moment of inertia

Figure 2.8 Geometry for astronaut arm kinematics.

In this case, the explicit form of equation (1) is

$$\begin{bmatrix} \dot{y}_e \\ \dot{z}_e \end{bmatrix} = \begin{bmatrix} -l_1 s_1 - l_2 s_{12} - l_3 s_{123}, -l_2 s_{12} - l_3 s_{123}, -l_3 s_{123} \\ l_1 c_1 + l_2 c_{12} + l_3 c_{123}, l_2 c_{12} + l_3 c_{123}, l_3 c_{123} \end{bmatrix} \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \end{bmatrix} \tag{2.7}$$

where,

$$\begin{aligned} s_1 &= \sin q_1, c_1 = \cos q_1, \\ s_{12} &= \sin(q_1 + q_2), c_{12} = \cos(q_1 + q_2), \\ s_{123} &= \sin(q_1 + q_2 + q_3), c_{123} = \cos(q_1 + q_2 + q_3), \end{aligned}$$

By inverting the Jacobian matrix, it is possible to calculate the arm joint velocities required to achieve a prescribed velocity of the capture bar. The inverse of the Jacobian matrix is given by,

$$J^{-1} = \frac{1}{l_1 l_2 s_2} \begin{bmatrix} l_2 c_{12} & , & l_2 s_{12} \\ -l_1 c_1 - l_2 c_{12}, & & -l_1 s_1 - l_2 s_{12} \end{bmatrix} \tag{2.8}$$

and the velocities are related by,

$$\dot{q} = J^{-1}\dot{p} \tag{2.9}$$

Integrating these velocities provided joint angle time histories that were subsequently used to drive the animation depicting the extension of the crewmember's arms to bring the capture bar into contact with the satellite's structural interface ring. The arm kinematics were related to a reach envelope analysis performed to determine how long it took the satellite to translate beyond the crewmember's reach.

Animation of the results was performed on a Silicon Graphics Indigo2 computer. Two computer animation programs, SSM (solid surface modeler) and OOM (object oriented modeler), developed at NASA's Johnson Space Center were used for this purpose.

Results of Intelsat Simulation

It was observed that the most significant perturbation of the motion of the satellite occurred for case C. Recall that this scenario modeled the interaction from the astronaut and capture bar as an unbalanced two-sided contact with a counter-rotation torque (from friction or mechanical interference). The resultant force acting parallel to the spin axis was about 85 N. The counter-roll (Y-axis) moment was -189 N-m and the yaw moment was 26 N-m. Not only did the interactive forces for case C agree most closely with the estimates of the EVA crewmembers, but the resulting motion of the satellite was seen in retrospect to correspond very closely with the motion observed in video footage of the actual EVA. For conciseness, only the results of the case C scenario are presented in this section.

A close up view of the computer graphics display for the Intelsat simulation is shown in Figure 2.9. The image shows the astronaut with his arms drawn back in the initial configuration at the start of the simulation. At this point, the satellite is in a stable spin with a rate of 1 rpm. During the first phase of the simulation, the astronaut extends his arms forwards until the capture bar comes into contact with the satellite, as shown in the first image of Figure 2.10. The subsequent images show the motion of the satellite resulting from the forces exerted by the astronaut through the capture bar. Large nutation angles are observed due to the loss of spin stabilization incurred by the reduction in spin rate resulting from the counter-rotary moment.

It was noticed that the animation of the Intelsat simulation, performed using the OOM program developed at NASA's Johnson Space Center, contained very noticeable jumps

between visual frames. It is believed that these discontinuities are a consequence of the highly detailed graphics used for the animation images. While the use of a very large number of polygons enhances the realism and attractiveness of the screen images, it severely compromises the computer image refresh rate. In developing the EVADS graphical display interface (described later in this thesis) it was decided that images should be simple, utilizing a relatively small number of polygons, so that real time animation would be smooth and life-like.

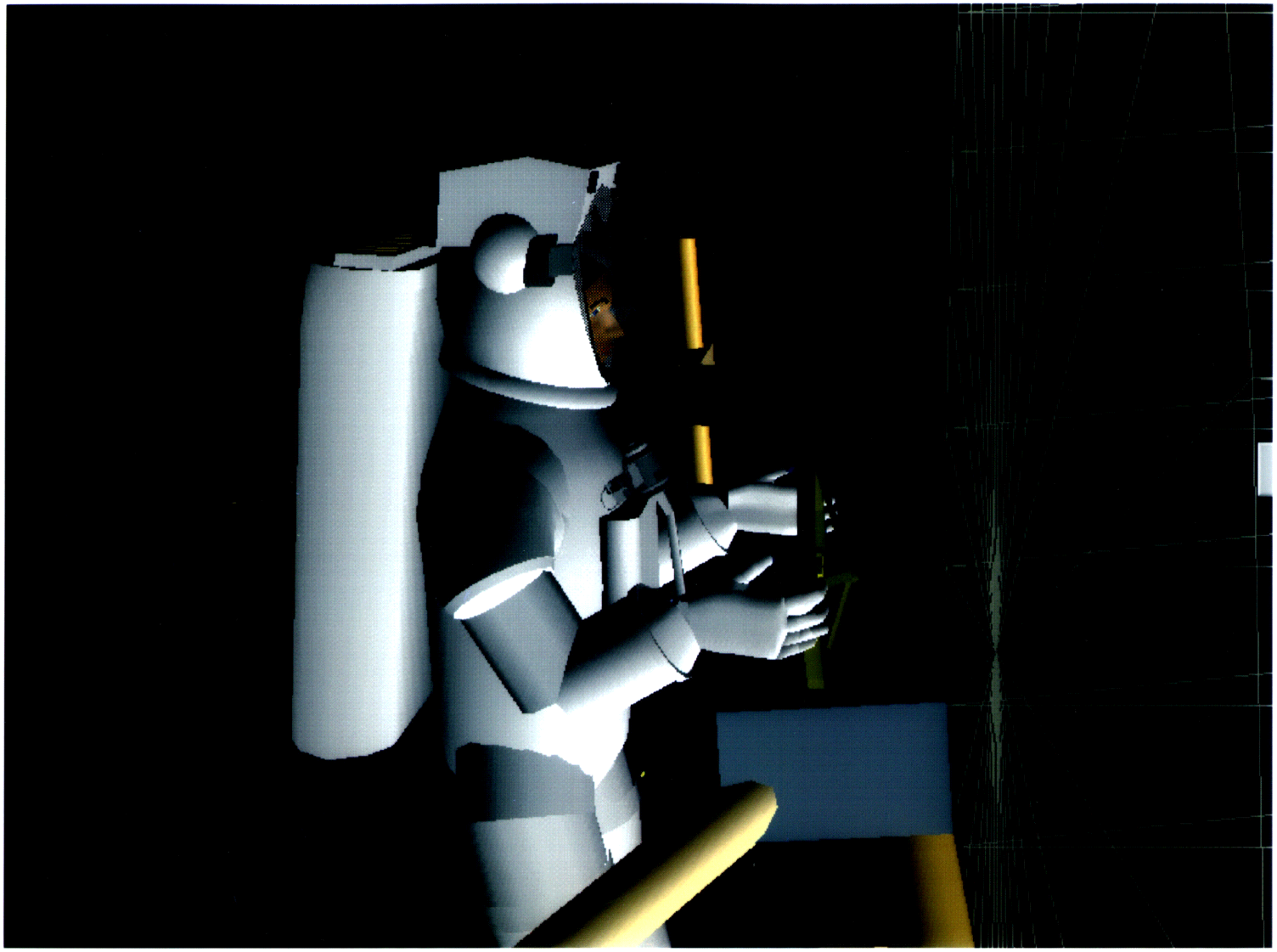


Figure 2.9 Close up view of astronaut and capture bar at start of Intelsat simulation. Note that arms are retracted in the ready position.

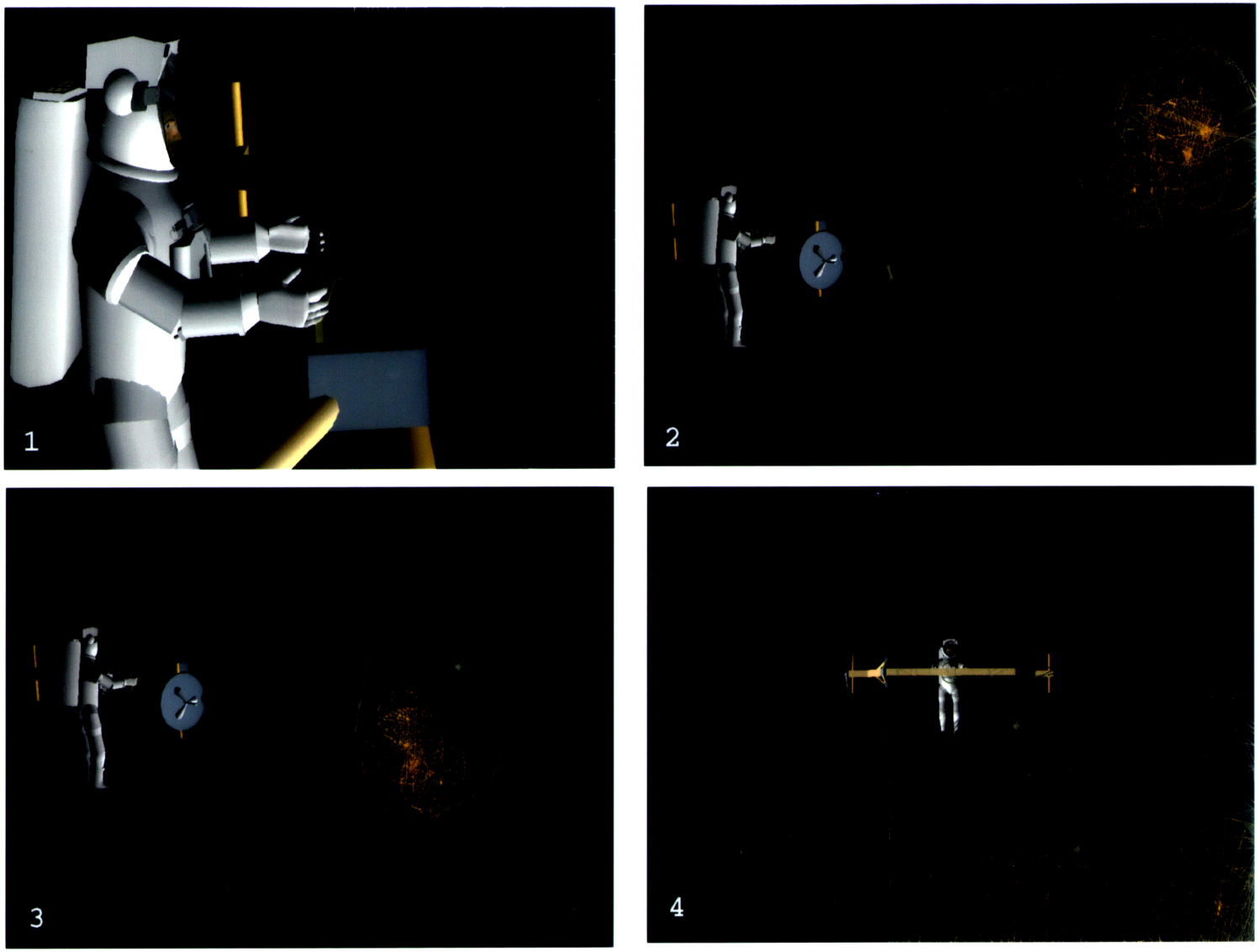


Figure 2.10 Animation sequence of Intelsat simulation for case C (unbalanced two–sided contact with counter–rotary moment).

The translational and rotational parameters of Intelsat, following capture bar interaction, are shown in the plots of Figure 2.11.

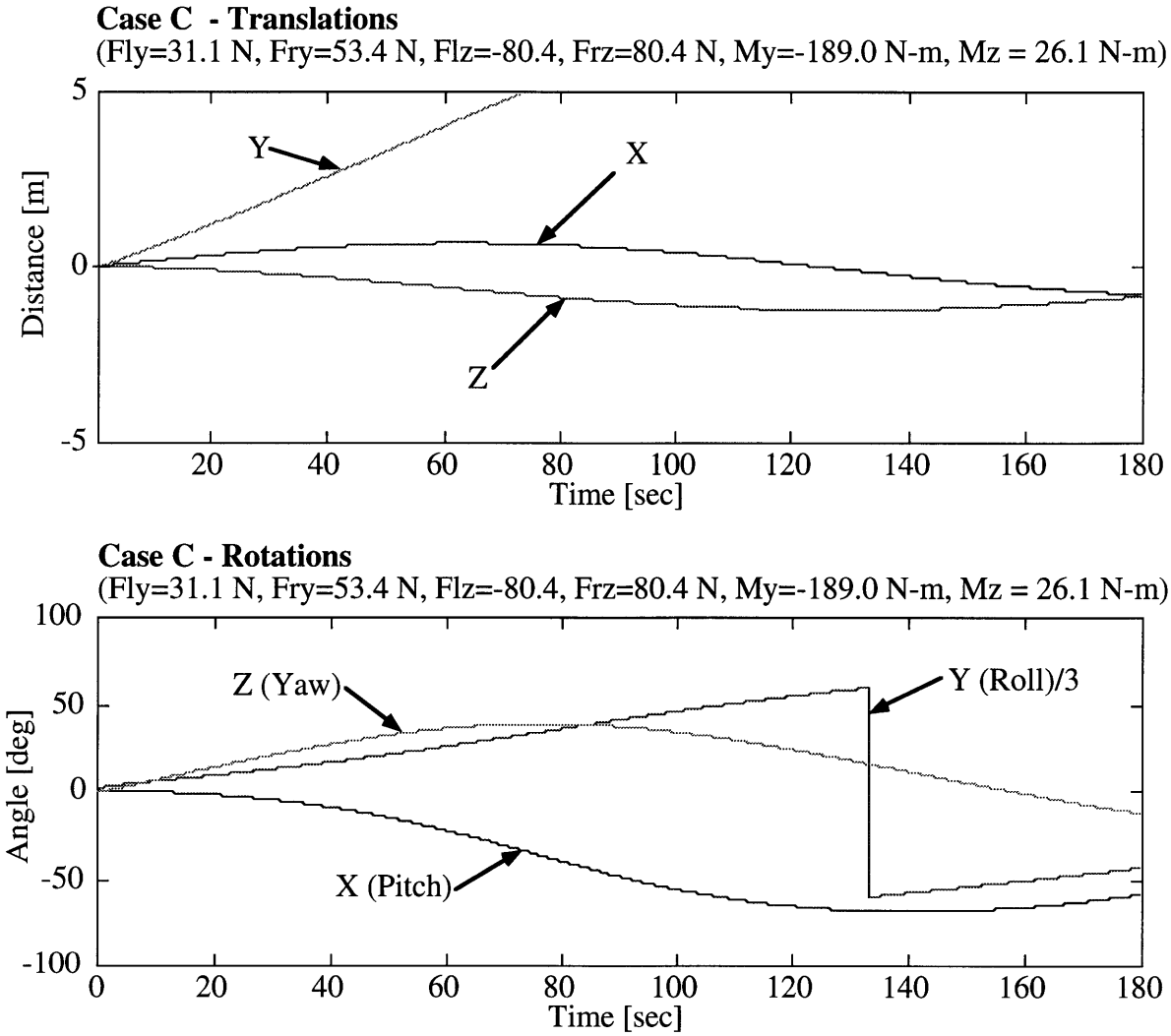


Figure 2.11 Translational (top) and rotational (bottom) motion of Intelsat VI satellite following Case C interaction (unbalanced two-sided contact with counter-rotary moment).

Linear displacements are shown in the upper plot of Figure 2.11. The X and Z translations exhibit sinusoidal fluctuations with maximum deviations of approximately 0.6 m in X and -1.3 m in Z. The Y displacement, however, continues to increase linearly with time, due to the constant velocity imparted by the axial force, reaching a value of 5 m at around 73 seconds.

Angular displacements are shown in the lower plot of Figure 2.11. After an initial reduction in roll rate, due to the counter-rotary moment, the roll angle continues to increase in a linear trend (with some barely observable fluctuations) at a rate of about 0.22 rpm. Note that the roll angle curve is plotted at one third scale to facilitate viewing

of the other two curves. Discontinuities in this curve are caused by a wrap around representation that produces the jumps from 180° to -180° (scaled down to 60° to -60°). The severity of the resulting nutations is revealed by the large pitch (X) and yaw (Z) angular displacements, the pitch angle reaching approximately -70° and the yaw angle reaching approximately 40°. Much smaller nutation angles (less than 16°) were observed in cases A and B since these scenarios did not involve a loss of spin stability associated with a change in angular momentum, as in case C.

Some important conclusions can be drawn from this analysis. The relatively low forces and moments expected from normal crewmember interaction with the satellite were destabilizing enough to cause large nutations in the satellite's motion. While such a state would prevent further attempts at capture, it would not be a serious issue if the satellite was successfully captured since the crewmember could damp out the nutations with his arms. Unfortunately, since the astronaut must exert an axial force on the satellite to keep the capture bar in contact with the satellite interface ring, and since EVA crewmembers estimate that this force cannot be less than about 44 N (10 lbf)[4] due to the limited tactility and proprioception allowed by the spacesuit, the satellite translates beyond reach in only 5-10 seconds. Such a short duration does not allow enough time for the trigger mechanism in the capture bar to reach the projections that cause the latches to fire. Thus, it is seen that the EVA crewmember was called upon to execute an apparently impossible task. It is believed that this difficulty did not surface during training because the small force levels that proved critical in space were masked by friction in the air bearing floor simulator.

2.5.4 Limitations and Challenges of Computational Simulation

It is important to acknowledge the limitations of computational simulations of multibody dynamic systems, especially for human body modeling. Simulations are limited by complexity, anatomical accuracy, control strategies, multiple solutions in inverse kinematics and dynamics, and lack of physical interaction (as compared to physical simulators).

Multibody dynamics simulations are inherently complex. This is due to the rapid increase in the number of equations of motion and size as more bodies and degrees of freedom are added to the system. Joints and interaction with external objects are accounted for by means of constraint equations, and these are usually not easy to implement. Due to this complexity, it is advisable to begin the analysis of a dynamic task

---

[4]Estimate based on discussions with EVA crewmembers.

by creating the simplest model possible, with the least number of degrees of freedom. Once this model is performing successfully, it can be expanded to account for more realistic situations. As an example of this, one might begin the analysis of an astronaut manipulating an object by considering only two arm segments and two degrees of freedom. The rest of the astronaut's body (trunk, upper legs, lower legs, etc.) could be incorporated progressively. Fortunately, it is often the case that an intuitive understanding of the physical situation being represented can suggest natural simplifications. For example, the torque in the astronaut's hip, knee, and ankle joints might be modeled by passive springs and dampers in an inverse dynamics simulation in which most of the motion occurs in the arm.

Since the emphasis of this research is on modeling the human body in dynamic EVA tasks, an important consideration is the accuracy with which models simulate human anatomy and kinematics. Clearly, the human body is far more complex than a system of simple rigid bodies connected by rudimentary pin, hinge, or ball and socket joints. In some joints, such as the elbow, knee, and ankle, multiple bones come together to form an interface where the center of rotation may describe a locus rather than a point. Also, the bones are most often held together by a complicated criss-cross pattern of ligaments with their own inherent elasticity, contributing to passive torques in the joints. Furthermore, due to the location of insertion points of muscles into bones, the relationship between muscle tension and muscle length and velocity is not linear. To further compound this, most joints are actuated by multiple muscle groups. Another factor is the mechanical properties of tissues in and around a joint, such as cartilage and skin. Obtaining accurate mass properties data (mass, moments of inertia, and products of inertia) for segments represents another obstacle. Current techniques are most often based on data obtained from cadaver measurements, although CAT and MRI imaging technologies hold more promise. One computer program that achieves a high level of accuracy in modeling human anatomy (movement is dealt with on a quasi-static basis) is SIMM (Software for Interactive Musculoskeletal Modeling) by Musculographics, Inc. in Evanston, IL. This program goes so far as to model joint centers of rotation, bone shapes, and ligament and tendon insertion points. SIMM is used primarily to predict the effects of orthopedic surgery and at this point is not well suited to dynamic simulation.

Human control strategy represents an area of study significant in itself. Some areas of focus include impedance control, neural networks, and optimal control (Pandy, Zajac et al. 1990). The research effort described in this thesis did not delve very deeply into control issues, although, other researchers in the laboratory are studying this topic and there are plans to incorporate human control models into EVA simulations in the future.

The type of simulation of most interest in analyzing EVA tasks is that of inverse dynamics, i.e., estimating the joint forces and torques required to perform a certain motion. For example, if an astronaut is required to manipulate a rather massive ORU (Orbital Replacement Unit) along a certain trajectory in a prescribed time and start and stop at certain locations, it would be of great interest to determine whether this task requires torques that exceed the range of human capability. It would also be useful to determine how a task should be performed to optimize human performance. Inverse kinematics and inverse dynamics analyses, however, encounter a problem in the case of dynamic systems with redundant degrees of freedom (i.e., more degrees of freedom than task coordinates), namely, multiple solutions to a given endpoint motion. To select a particular solution in such cases, one might employ a technique such as linearized least squares or optimization. Care must be taken, however, to ensure that the solutions obtained make physical sense, such as verifying that none of the joints are hyperextended.

One last limitation worth discussing, in relation to the desire to use these simulations for astronaut training, is the lack of physical interaction in a computer simulation. Recent advances in Virtual Reality technology make it possible to address this issue, particularly when force feedback and haptic sensors and actuators are employed. Virtual Reality is already used in astronaut training for EVA, although primarily as a way of familiarizing the astronauts with the visual environment and practicing communication protocols. Combining Virtual Reality with dynamic simulation in the future will undoubtedly provide astronaut training personnel with a powerful and versatile tool.

3. Methodology

This chapter describes how computational multibody dynamics can be applied to the analysis and simulation of EVA tasks. The first goal is to familiarize the reader with the process of multibody dynamical equation formulation and to demonstrate the complexity of these equations. A detailed example of this process as formulated by hand for a relatively simple system presents the equations in an explicit form. For clarity and convenience, the Lagrangian method is used for this first example. The remainder of the chapter describes how computational methods are employed to create computer simulations of more complex systems. The main steps are: the creation of a system description file; employing SD/FAST to derive the dynamical equations and simplify them using symbolic manipulation; development of user-written simulation driver code; and analysis of results by means of animation and plots. To illustrate these steps, a particular simulation example is presented based on an actual EVA mass handling task recently performed in space. In this case, the equations of motion are highly complex and would take several pages to write out. This would be extremely tedious to perform and would defeat the purpose of representing the equations implicitly by means of computer code and performing the analysis by numerical means. This chapter focuses on explaining how the system description file is created and how the simulation and analysis is carried out, while the interim step of generating the equations of motion is the domain of the commercially available program SD/FAST. Some general points on the SD/FAST computations were presented in the previous chapter. For more details on how the equations of motion are formulated, the reader is referred to additional references (Hollars, Rosenthal et al. 1994) (Kane, Likins et al. 1983) (Kane and Levinson 1985) (Rosenthal and Sherman 1986).

3.1 Example of Dynamical Equation Formulation

A relatively simple multibody system example has been chosen to ensure readability. The example is a two-body, two-d.o.f model of an astronaut's arm performing a manipulation task. It has been assumed that the rest of the astronaut's body has no influence on the dynamics of his arm motions because his torso is fixed in inertial space. As highly simplified as the scenario sounds, there is a conceivable situation in which the astronaut's backpack is affixed to an object with a much larger mass, such as the Space Shuttle Orbiter. The inertia of such a massive object would constrain the accelerations of

the astronaut's trunk to insignificant values allowing one to make the approximation that the trunk is stationary in inertial space. To further simplify the model, the arm is restricted to planar motion and only has two degrees of freedom: shoulder planar rotation and elbow joint extension and flexion. The upper and lower arm are modeled as rigid segments with constant moments of inertia. The hand is considered to be attached to the center of gravity of the object being manipulated. This model is sketched in Figure 3.1.

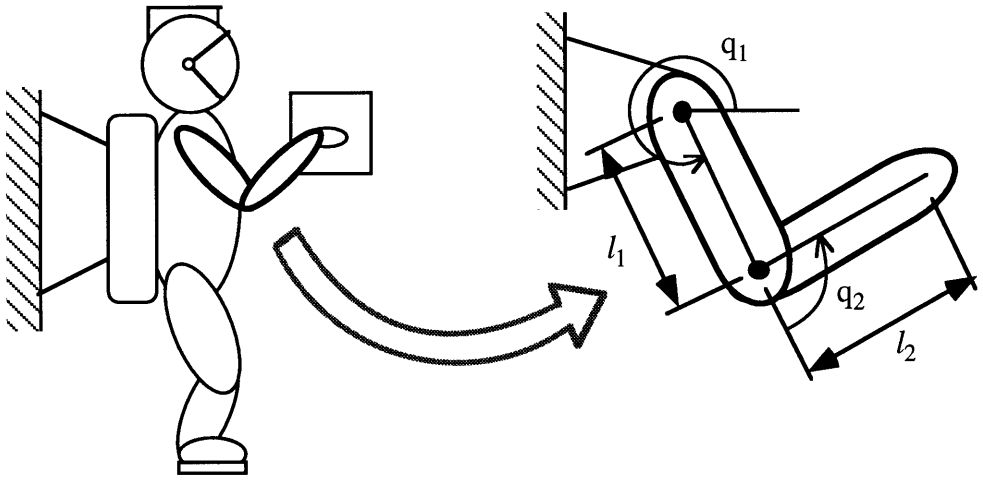


Figure 3.1 Two d.o.f. model of astronaut arm.

As mentioned previously, the two most common classical formulations of multibody dynamics are the Newton-Euler formulation and the Lagrangian formulation. This astronaut arm motion example derives the equations of motion by means of the Lagrangian formulation in which the behavior of a dynamic system is described in terms of work and energy stored in the system. The advantage of the Lagrangian approach is that the constraint forces involved in the system are automatically eliminated, as opposed to the manual elimination required in the Newton-Euler method. Furthermore, the closed-form dynamic equations can be derived systematically regardless of the coordinate system chosen. The derivation given below is based on the description of Lagrangian dynamics given in the text *Robot Analysis and Control* (Asada and Slotine 1986).

3.1.1 Lagrangian Formulation

If $q_1, \ldots, q_n$ are the generalized coordinates describing the orientation of a system, and T and U are the total kinetic energy and potential energy respectively, then the Lagrangian L is defined by

$$L(q_i, \dot{q}_i) = T - U \tag{3.1}$$

The equations of motion of the system are given by

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{q}_i} - \frac{\partial L}{\partial q_i} = Q_i \qquad i = 1, \ldots, n \tag{3.2}$$

where Q_i represents the generalized force that corresponds with q_i, the joint angle coordinates.

Ultimately, the objective is to obtain a formula for calculating the inertia tensor of the system. First, however, the Jacobian matrices relating the geometry of the segments and their joint centers are expressed in the form

$$\begin{aligned} J_L^{(i)} &= \left[J_{L1}^{(i)} \ldots J_{Li}^{(i)} \;\; 0 \ldots 0 \right] \\ J_A^{(i)} &= \left[J_{A1}^{(i)} \ldots J_{Ai}^{(i)} \;\; 0 \ldots 0 \right] \end{aligned} \tag{3.3}$$

Each column vector is given by

$$J_{Lj}^{(i)} = \begin{cases} b_{j-1} & \text{for a prismatic joint} \\ b_{j-1} \times r_{0,ci} & \text{for a revolute joint} \end{cases}$$

$$J_{Aj}^{(i)} = \begin{cases} 0 & \text{for a prismatic joint} \\ b_{j-1} & \text{for a revolute joint} \end{cases} \tag{3.4}$$

where j represents the the column number in the Jacobian matrix, b_{j-1} is the 3 x 1 unit vector representing joint axis j-1, and $r_{0,ci}$ is the position vector to the centroid of segment i with reference to the base coordinate system. A prismatic joint allows only translational relative motion and a revolute joint only allows rotational relative motion between connected segments.

The total kinetic energy stored in the system is given by

$$T = \frac{1}{2}\sum_{i=1}^{n}\left(m_i \dot{q}^T J_L^{(i)T} J_L^{(i)} \dot{q} + \dot{q}^T J_A^{(i)T} I_i J_A^{(i)} \dot{q} \right) = \frac{1}{2}\dot{q}^T H \dot{q} \tag{3.5}$$

and is expressed in terms of the joint velocities $\dot{q} = \left[\dot{q}_1, \ldots, \dot{q}_n \right]^T$ that are the derivatives of the joint displacements $q = \left[q_1, \ldots, q_n \right]^T$ that represent a complete set of generalized

coordinates. The matrix H is known as the *manipulator inertia tensor*[5] and contains the mass properties of the complete arm linkage. This n x n matrix is obtained from the formula

$$H = \sum_{i=1}^{n} \left(m_i J_L^{(i)T} J_L^{(i)} + J_A^{(i)T} I_i J_A^{(i)} \right) \tag{3.6}$$

The manipulator inertia tensor is a symmetric positive definite matrix whose quadratic form relates to the kinetic energy. It is important to note that, since Jacobian matrices are involved, which vary with the orientation of the arm, the manipulator inertia tensor is configuration-dependent and represents the combined mass properties of the whole system for a given configuration. A consequence for computer simulations is that the terms of the matrix must be recalculated for each time interval during the execution of a movement, and this can significantly increase processing time.

The next step is to consider the generalized forces $Q = [Q_1, \ldots, Q_n]^T$ that account for all the forces and moments acting on the arm linkage other than the inertial and gravity forces. The generalized forces are identified as

$$Q = \tau + J^T F_{ext} \tag{3.7}$$

where τ represents the joint torques and F_{ext} represents the external forces acting on the system. Finally, the equations of motion are obtained from the expression

$$\sum_{j=1}^{n} H_{ij} \ddot{q}_j + \sum_{j=1}^{n} \sum_{k=1}^{n} h_{ijk} \dot{q}_j \dot{q}_k + G_i = Q_i \qquad i = 1, \ldots, n \tag{3.8}$$

where

$$h_{ijk} = \frac{\partial H_{ij}}{\partial q_k} - \frac{1}{2} \frac{\partial H_{jk}}{\partial q_i} \tag{3.9}$$

and

---

[5]Derived from robotic systems terminology.

$$G_i = \sum_{j=1}^{n} m_j g^T J_{Li}^{(j)} \tag{3.10}$$

Note that H_{ij} represents the term or expression in the i-th row and j-th column of H. The first term in equation 3.8 accounts for the inertial torques, the second term represents the centrifugal and Coriolis effects, and the third term is the torque due to gravity. This formulation directly provides the closed-form dynamics equations.

3.1.2 Equations of Motion for a Two Degree of Freedom Arm

When the general Lagrangian equations given above are applied to the specific case of the two segment, two degree of freedom arm illustrated in Figure 3.1, the following equations of motion are obtained:

$$\begin{aligned} H_{11}\ddot{q}_1 + H_{12}\ddot{q}_2 + h_{122}\dot{q}_2^{\,2} + (h_{112} + h_{121})\dot{q}_1\dot{q}_2 &= \tau_1 \\ H_{22}\ddot{q}_2 + H_{12}\ddot{q}_1 + h_{211}\dot{q}_1^{\,2} &= \tau_2 \end{aligned} \tag{3.11}$$

The manipulator inertia tensor is given by

$$\mathbf{H} = \begin{bmatrix} m_1 l_{c1}^2 + I_1 + m_2(l_1^2 + l_{c2}^2 + 2l_1 l_{c2}\cos q_2) + I_2 & m_2 l_1 l_{c2}\cos q_2 + m_2 l_{c2}^2 + I_2 \\ m_2 l_1 l_{c2}\cos q_2 + m_2 l_{c2}^2 + I_2 & m_2 l_{c2}^2 + I_2 \end{bmatrix} \tag{3.12}$$

where

$$\begin{aligned} &h_{122} = -h \;,\; h_{112} + h_{121} = -2h \;,\; h_{211} = h \;, \\ &\text{and} \quad h = m_2 l_1 l_{c2}\sin q_2 \end{aligned} \tag{3.13}$$

The lengths l_{c1} and l_{c2} are measure from the joint center to the corresponding segment's center of mass. Note that the gravity terms normally appearing in the equations of motion have dropped out due to the weightless environment. Also, only joint torques appear on the right hand side of the equations of motion (3.11) since it has been assumed that there are no external forces acting on the arm.

Fairly complex equations of motion are seen for a relatively simple dynamic system. As more segments and more degrees of freedom are added, not only do the individual elements of the manipulator inertia tensor become longer expressions, but also the number of elements increases according to the square of the number of degrees of freedom. Clearly, it soon becomes impractical to formulate the equations of motion by

hand. It is for this reason that so much effort has been invested in developing computer programs that formulate the equations computationally. The simulation methods and results presented in the remainder of this thesis take advantage of, and build upon, powerful computational techniques that have been developed for the purpose of analyzing more complicated dynamic systems.

3.2 STS-63 Spartan Mass Handling EVA Task

Before delving into the details of how the simulations are performed, it is helpful to consider a brief description of the actual EVA on which the simulations were modeled. The Spartan spacewalk, performed on Space Shuttle mission STS-63 in March 1995, had two primary objectives: firstly, to test the use of thermal insulation in the EMU gloves in cold attitude operations (crewmembers in shadow) and to gain experience with handling large masses. It is the second objective that serves as the subject task for simulation.

A free flying payload, the Spartan 204 astronomy spacecraft, served as the mass handling test object for the EVA crewmembers. A NASA photograph image of the astronauts with Spartan is shown in Figure 3.2. Although the Spartan 204 was designed to be deployed and berthed by means of the Orbiter's Remote Manipulator System (RMS), or robot arm, a contingency EVA was planned in which manual berthing of the payload could be achieved in case of a failure of the nominal RMS berthing procedure. Contingency EVAs are often planned, and trained for, with free flying payloads of this type, although, other than the opportunities presented during the Hubble space telescope repair mission, little experience has been gained in the manipulation of objects with significant inertial properties.

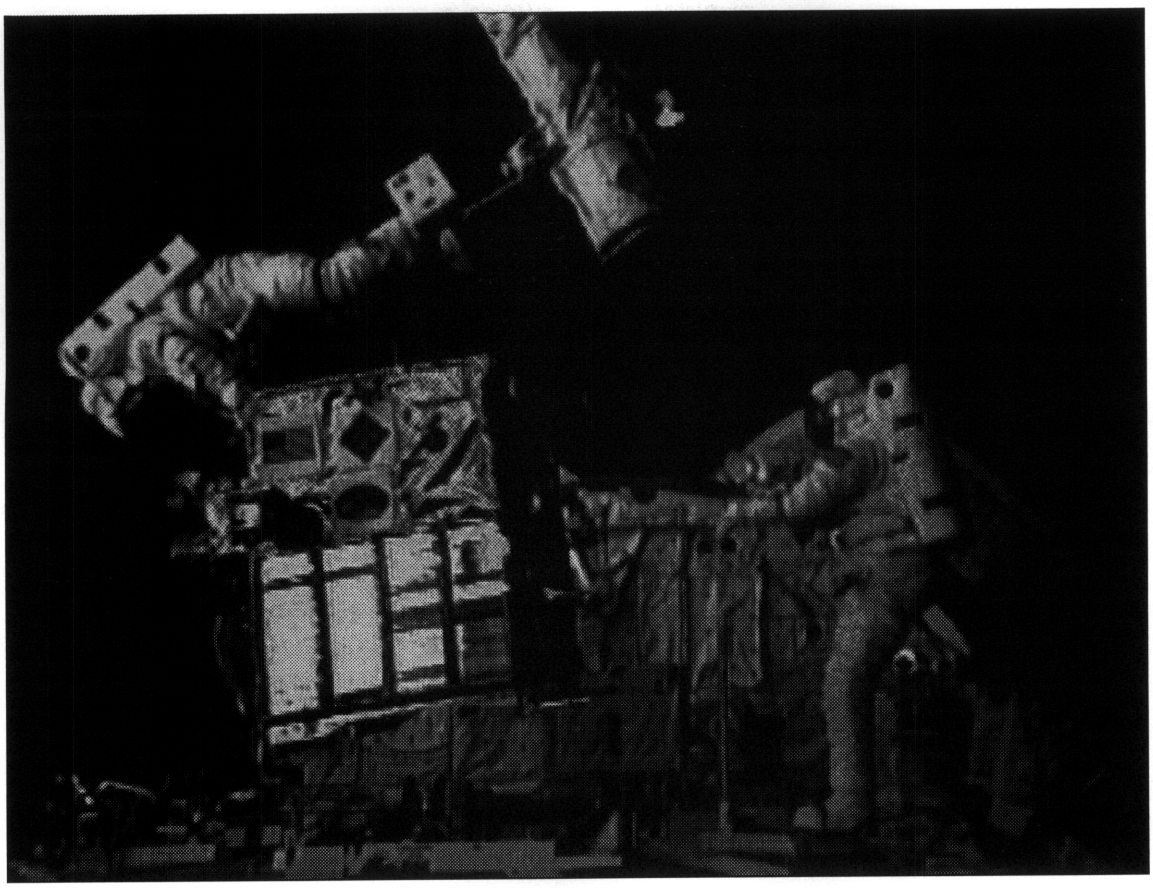


Figure 3.2 EVA crewmembers with Spartan 204 free flyer payload. (Source: NASA)

In addition to the contingency procedure, it was proposed that Spartan could be used for a scheduled EVA in which crewmembers could practice handling this massive object as a way of preparing for similar mass manipulation tasks to be performed very often during the construction and servicing of the international space station. The mass properties for the Spartan free flyer are presented in Table 3.1.

Table 3.1 Mass property data for Spartan 204 free flyer.

Mass: 1,201.07 kg (82.30 slugs)	**Moments of Inertia:**
	I_{xx} = 325.73 kg-m^2 (240.31 slug-ft^2)
Center of Mass (PAS†):	I_{yy} = 352.28 kg-m^2 (259.90 slug-ft^2)
X = 0.622 m (2.04 ft)	I_{zz} = 334.79 kg-m^2 (246.997 slug-ft^2)
Y = -0.481 m (-1.58 ft)	
Z = 0.572 m (1.88 ft)	**Products of Inertia:**
	I_{xy} = -3.75 kg-m^2 (-2.77 slug-ft^2)
	I_{xz} = 92.67 kg-m^2 (68.37 slug-ft^2)
	I_{yz} = -28.34 kg-m^2 (-20.91 slug-ft^2)

† PAS = Payload Axis System, defined with respect to the bottom left corner on the thermal louver side.

The most striking value in Table 3.1 is the Spartan payload's mass, which is more than fifteen times the mass of an average crewmember (not including EMU). Clearly, this represents a far greater mass than anything a person might have experience handling in a one-g environment.

While it is true that in a weightless condition one can theoretically move objects of unlimited size, any relative kinetic energy that the crewmember imparts to the object must similarly be removed by the crewmember if he wishes to keep hold of it. The crewmember could find himself trapped between two objects (i.e., the payload and the RMS or Orbiter), or forced into limb hyperextension, resulting in bodily injury or damage to the spacesuit, if he does not anticipate the object's motion.

It is difficult for astronauts and mission operations personnel to predict, at any quantitative level, the types of loads that might be experienced in handling large objects in weightlessness, both because of the lack of empirical data on this procedure and lack of analytical models of the multibody dynamics involved. The prime objective of this study is to demonstrate how a computer program, which combines the mathematical rigor of computational multibody dynamics and the communicative strengths of animation and

data plots, might be used to predict the quantitative and qualitative aspects of human performance in extravehicular activity. The sections below describe how this objective was pursued.

3.3 Simulation Objectives

The particular scenario that is modeled during these simulations is that of an EVA crewmember manipulating the Spartan 204 free flyer along a particular trajectory at constant speed. A circular trajectory of radius 0.15 m with an angular velocity of 0.628 radians per second (one complete revolution in ten seconds) is prescribed for the center of mass of the crewperson's hand. The orientation of the Spartan payload remains fixed with no angular velocity being imparted. Translations are confined to the vertical plane. It is assumed that the astronaut maintains a rigid grip on a handle attached to Spartan.

It is necessary, at this point, to distinguish between the terms *endpoint coordinates* and *joint coordinates*. The relationship between the two systems is illustrated in Figure 3.3. Endpoint coordinates refers to the description of the position and orientation of the end-effector of a multibody chain dynamic system (typically a robot arm or a human limb). Usually, an endpoint coordinate system is specified at a particular point on the last body in the chain. The position and orientation of the coordinate system is described in reference to the global coordinate system, usually fixed in an inertial reference frame. Joint coordinates, on the other hand, describe the state of the multibody system in terms of the angular orientations (or linear displacements in the case of sliding joints) of the bodies, with respect to either the inboard body or the global coordinate system (called "generalized" coordinates). The two coordinate systems are related through the Jacobian matrix of the system.

There is an important difference between the two coordinate systems. While the joint coordinate system always describes the state (configuration) of a system uniquely, a description of the state of a system in terms of endpoint coordinates may not specify the system's configuration uniquely if there are redundant degrees of freedom. A redundant degree of freedom condition arises when there are more joint coordinates than endpoint coordinates. The problem introduced for inverse kinematics is that there may be multiple configurations (or states) of a system producing a given endpoint state. At the same time, in terms of inverse dynamics, this means that there may be different joint torque (and force) solutions that satisfy the given end-effector forces and torques. The geometric ambiguity is shown by means of the dashed outline in Figure 3.3 (Configuration 2).

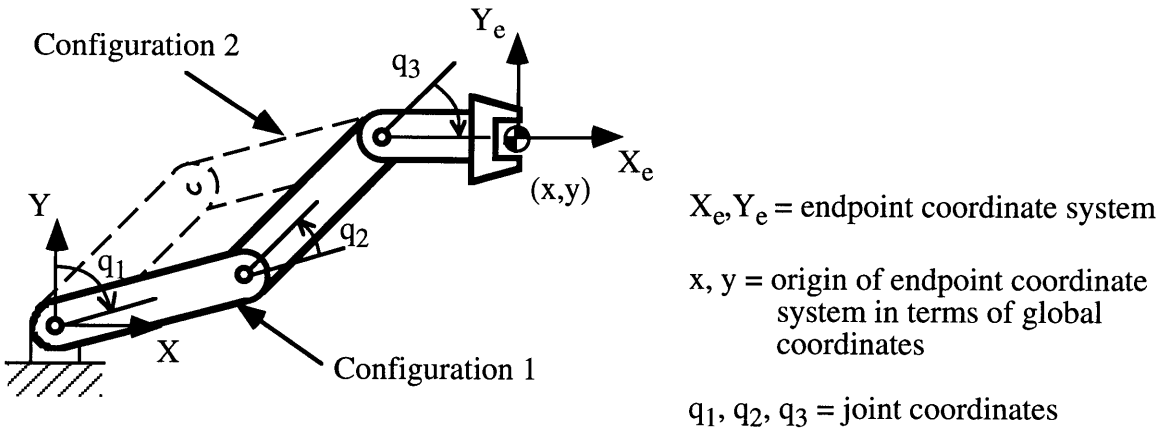


Figure 3.3 Illustration of joint coordinates and endpoint coordinates and occurrence of multiple solutions in systems with redundant degrees of freedom.

It is particularly desirable to be able to specify the motion to be performed by the crewmember in terms of endpoint coordinates only, that is in terms of the X and Y positions of the c.m. (center of mass) of the hand. Describing the motion in this manner demonstrates that it is possible to perform a simulation by knowing only the motion data associated with the task itself, without the need to explicitly prescribe the motion being performed in terms of joint coordinates. Only the initial angles for the wrist, elbow, shoulder, hip, knee, and ankle joints need be known. The subsequent time histories of position, velocity, acceleration, and torque for these joints are calculated during the simulation. The importance of this capability is that it allows analysts, astronaut trainers, or mission operations personnel, to describe the parameters of an EVA task to be simulated in a simple pragmatic way and does not require them to do extensive analysis beforehand.

Thus the objectives of the dynamic simulation may be summarized as follows:

1) Determine the kinematics (in joint coordinates) of the crewmember's motions given only a description of the manipulation task (in endpoint coordinates).

2) Perform an inverse dynamics computation to determine the joint torques in the ankle, knee, hip, shoulder, elbow, and wrist.

3) Compare the calculated values with empirical human physiological limit data such as joint angle limits and maximum torque availability based on joint position and velocity.

4) Demonstrate how accuracy and realism can be improved in successive simulations.

3.4 Dynamic Simulation

The purpose of this section is to describe in greater detail how the simulation is actually carried out on the computer. Various phases may be identified: the creation of a system description file; formulation of the equations of motion using SD/FAST; development of simulation code, including steps to perform a validation of the model based on test values of joint torque, followed by inverse kinematics and inverse dynamics; comparison with physiological limits; and animation with data plots.

3.4.1 System Description

The first step in the development of a simulation is to develop a model of the multibody dynamic system under consideration. It is highly advisable to design the simplest possible model which is capable of satisfying the objectives of the simulation. Starting with a simple model facilitates the steps of error elimination and model validation in the early stage of a simulation. Once the rudimentary system model is operating correctly, one can expand the complexity of the model incrementally while verifying the validity of the model each time it is changed. Due to the inherent complexity of multibody dynamics, this turns out to be a very wise philosophy in practice.

In considering the structure of the human body, one might be inclined to think that the simplest model should include fourteen segments: two feet, two lower legs, two upper legs, a torso, a head, two upper arms, two forearms, and two hands. Certainly, models of much greater complexity can be imagined. For example, if the full articulation of the hands are to be modeled, then 19 degrees of freedom must be incorporated for each hand. However, it is also possible to use simpler models of the human body depending on the degree of localization of motion and the amount of detail required. For instance, if a task is accomplished almost exclusively by means of arm motions, then one might be able to assume that the torso is fixed in inertial space (thus becoming the "ground" segment) and model the arm simply as a three segment system. In addition, symmetry can often be exploited in both the model and the simulation. If a manipulation task is carried out in

which both arms perform the identical motion, then it may be possible to model both arms as a single arm, that is modeling the six segments as only three, but with double the mass properties of each segment. Once the torques have been determined, they may be divided in half to yield the contribution of each separate arm. This strategy relies on the assumption that mass properties and torques may be combined in a linear fashion when they occur in parallel.

Since the task being analyzed involves the manipulation of an object along a trajectory confined to the median plane of the crewmember, the limbs on the left and right sides of the body perform identical motions. In addition, it is assumed that the feet of the astronaut are rigidly fixed to a body of large inertia and thus can be considered to be part of the "ground" segment and not part of the dynamic system under consideration (A foot restraint would serve this purpose in practice; although a more advanced model might consider the compliance of the foot restraint, the RMS, and perhaps the Orbiter too.). Given these simplifications, it is possible to model the astronaut's body by means of seven segments with the Spartan 204 spacecraft as an eighth segment "welded" to the hand. A sketch of this system is shown in Figure 3.4. The segments are: lower leg, upper leg, torso, head, upper arm, forearm, hand, and Spartan.

To make it possible to specify the motion in terms of endpoint coordinates, the model makes use of an interesting trick. The explanation of this is aided by Figure 3.4. Up to this point, the crewmember's body is represented by a tree structure (no loop joints). In fact the articulated bodies, from the lower leg connected to ground, to the hand as the outermost body, represent a simple chain-link structure. The hand, however, is defined with half the density, and thus half the mass properties, of the intended hand for reasons that are explained below. An additional tree is now defined. Starting at ground (ankle joint), a massless body is created with an offset equal to the distance between the ankle and the center of mass of the hand and "attached" to ground by means of a slider in the X-direction. Another massless body is defined and attached to the first massless body by means of a slider in the Y-direction. This massless body is then pinned (axis in the Z-direction) to another body with half the mass and moments of inertia of the intended hand. This half-hand is welded to the half-hand attached to the fore-arm, thus completing a loop structure. The resulting hand has the mass and moments of inertia of a full hand. In this way it is possible to manipulate the arm and body of the crewmember in the X-Y plane by prescribing the motion of the center of mass of the hand in global Cartesian coordinates in terms of the sliding displacements of the two massless bodies.

The next step is to create a system description file. It is advisable to store this file in a separate directory bearing the name of the simulation. This directory will then also

contain the files generated by SD/FAST, the simulation code, and the output data files. The filename itself will be of the form <file>.sd, where <file> is chosen to be representative of the simulation and serves as the root for other files generated by SD/FAST. The system description file is simply a means of conveying to SD/FAST the relevant parameters and geometry of the system so that the equations of motion can be formulated. Full details on how this input file should be created can be found in the SD/FAST user's manual (Hollars, Rosenthal et al. 1994). The system description file for the eight segment model used in this simulation is presented in its entirety in Appendix A. A summary of the basic elements of this file will be presented here with examples taken from the actual system description file used in the simulation.

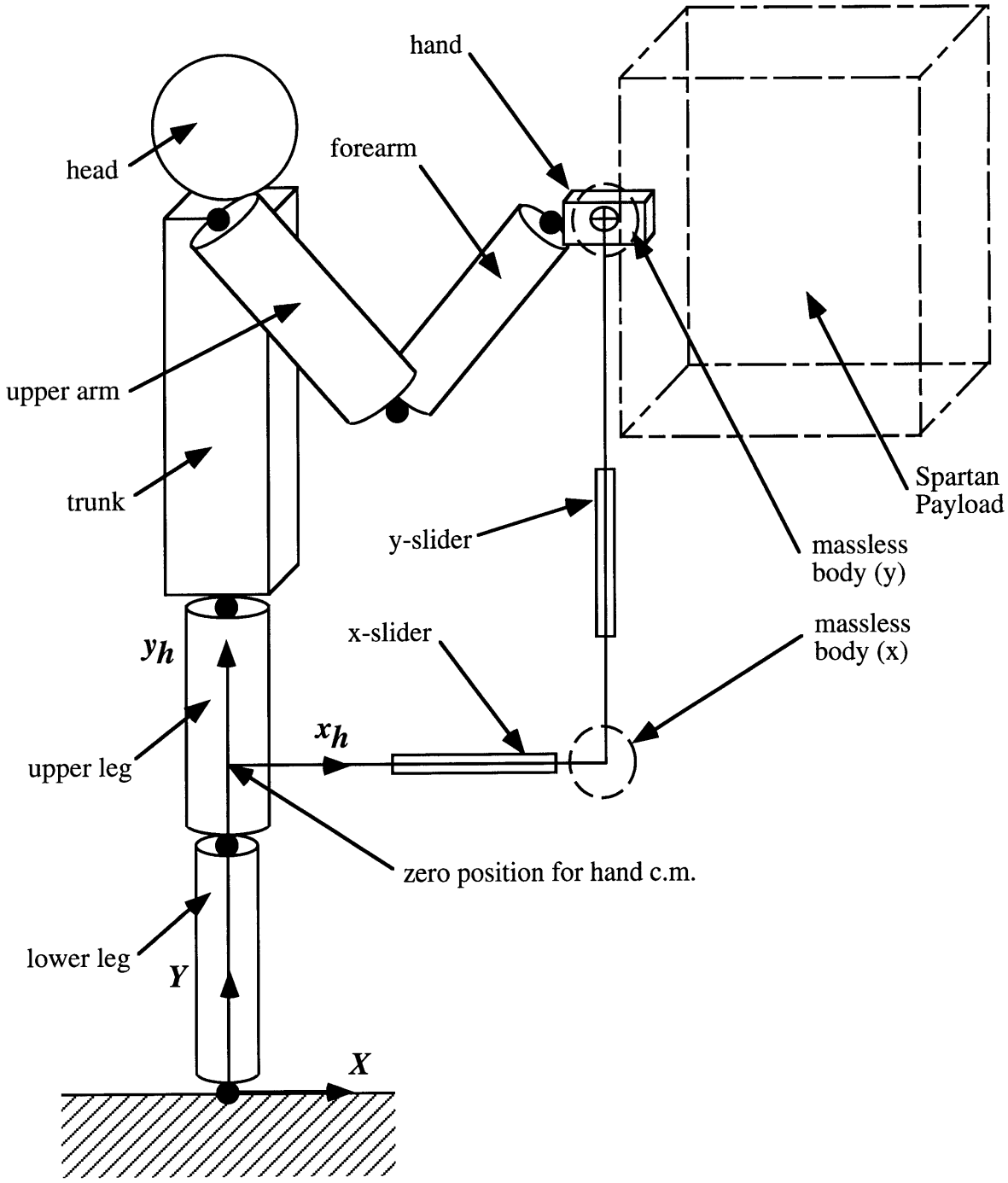


Figure 3.4 Sketch of eight segment system showing massless bodies and sliding joints for prescribing motion of center of mass of hand.

The first part of the system description file consists of comment lines which normally indicate the name of the file, the author, date of last revision, and a short description of the nature of the system. Other comment lines are added at appropriate points in the body of the file. The second part is normally a "preamble" which includes any keywords relevant to the entire system. An example of this would be the specification of gravity, e.g., "gravity = 0 -9.8 0", but in this case gravity has been zeroed out since the situation being simulated is assumed to occur in weightlessness. The third major part of the file consists of one or more "body paragraphs". Each of these specifies the relevant parameters of a body and how it is connected to other bodies in the system. All vectors and mass properties for the bodies are specified according to a reference configuration. It is advisable to choose a reference configuration that makes it as simple as possible to describe the vectors and mass properties. Wherever possible, the principal axes of each body should be aligned with one or more axes of the global coordinate system. The reference configuration for this simulation is shown in Figure 3.5. An example body paragraph is given below

```
body = uleg        inb = lleg      joint = pin      prescribed = ?
    mass = 17.300          inertia = .294244 .055360 .294244
    bodytojoint = 0 -.215 0        inbtojoint = 0 .215 0   pin = 0 0 1
```

The first keyword "body" is followed by the name of the body, "uleg" for upper-leg in this case. Next the keyword "inb" identifies the inboard body in the structure, which in this case is "lleg" for lower-leg. One of the bodies in the system must be connected to ground in some way and this is accomplished by assigning "$ground" to "inb". Next comes the joint specification. In the example above, this is simply a one d.o.f pin joint. The specification "prescribed = ?" notifies SD/FAST that prescribed motion may be used as an option for articulating this body. The question mark makes it possible to turn prescribed motion on and off from within the simulation driver code. The second line in this example specifies the mass and inertial properties of the body in SI units. If only three values follow the keyword "inertia" then it is assumed that these are the principal moments of inertia. It is possible, however, to specify a full 3 x 3 inertia tensor for an asymmetrical body. The last line contains three vectors describing the geometric state of the body when the system is in its initial reference configuration. (The reference configuration for this system, shown in Figure 3.5, has the astronaut standing up straight with his arms hanging straight down next to his sides.)

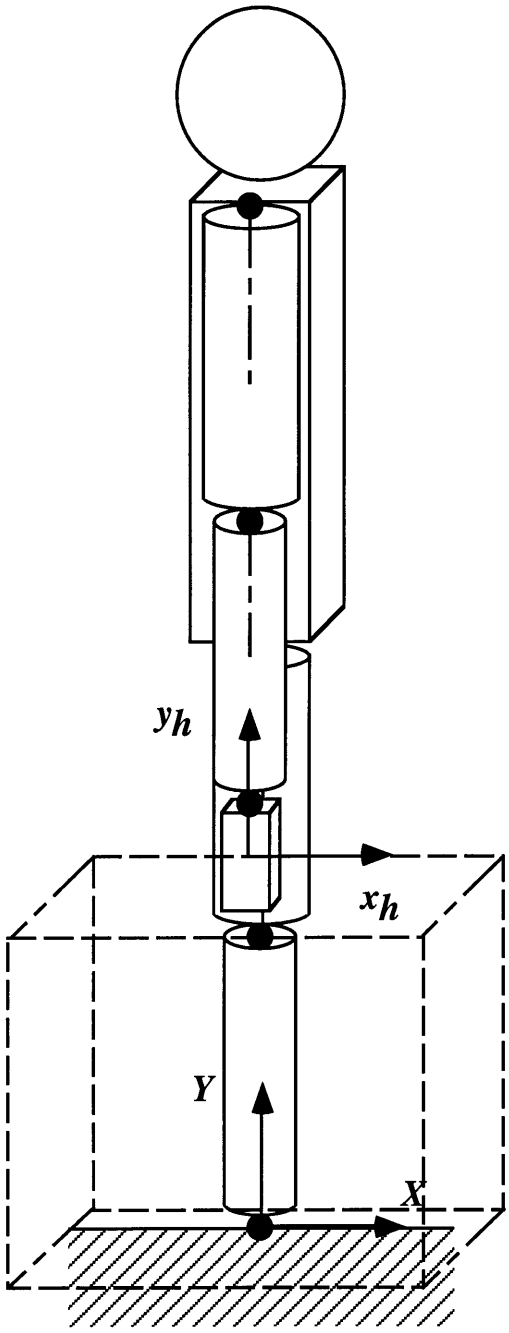


Figure 3.5 Reference configuration for description of dynamic system.

The first vector, "bodytojoint", extends from the body's center of mass to the joint connecting it to the inboard body. The second vector "inbtojoint" extends from the center of mass of the inboard body to the same joint. The last vector describes the direction of the joint axis (the axis of rotation, in the case of a pin or revolute joint, or the axis of translation for a sliding joint). Only the direction of this vector matters, so it is common to present it in a normalized form. Several of these body paragraphs make up the bulk of the system description file. Most of them follow the same format described above, however, one of these paragraphs is slightly different. This paragraph is

```
body = hslide      inb = harm     joint = weld
    bodytojoint = 0 0 0    inbtojoint = 0 0 0
    pin = 0 0 1            bodypin = 0 0 1
    inbref = 0 1 0         bodyref = 0 1 0
```

The code given above does not represent an actual body, but is used to tell SD/FAST how to weld the two halves of the hand together. The vectors "pin", which must align with "bodypin", and "inbref", which must align with "bodyref" and is perpendicular to the "pin" vector, are used to ensure that the two parts are welded together in the appropriate relative orientation. Since the weld joint connects two tree structures together, that originate from the same base (ground), it constitutes a "loop" joint.

3.4.2 Formulation of Equations of Motion

Once the dynamic system has been fully specified in the system description file, the next step is to process the system description file using SD/FAST. This step is a fairly easy one for the analyst. SD/FAST is invoked by simply typing

```
sdfast -lc
```

at the UNIX shell prompt. The "-lc" option tells SD/FAST to generate code in the C language. The name of the system description file can either be specified in the command line or entered when prompted by SD/FAST. If no syntactical errors are found and the input file is successfully processed, several new files are created: a Dynamics File (identified by <file>_dyn.c) that contains subroutines (or "functions" for C code) specific to the dynamic system and representing the equations of motion numerically; an Information File (identified by <file>_info) containing text and various parameters of use in creating the simulation driver code; and an Analysis File (identified by <file>_sar.c)

containing simple analysis routines that can be called during simulation runs. When the simulation driver code has been written, it is compiled and linked along with the Dynamics File and the Analysis File. Once again, a more detailed description of this step can be found in the SD/FAST user's manual. The next three sections describe the three major parts of the simulation driver code which is presented in its entirety in Appendix B.

3.4.3 Joint Torque Test Functions

The first step in the simulation is to verify the validity and accuracy of the dynamic model by executing three test functions which return torque values for the shoulder, elbow, and wrist. Only the arm joints are tested since it is the arm motions and torque values that are of primary interest in this simulation. By choosing a trivial configuration and simple motion, it is possible to calculate torque values for each of the joints by hand and these are compared with the values returned by the computer simulation.

The initial configuration of the body is with the lower leg, upper leg, and trunk straight up, as in the reference configuration, but with the arm straight out in a horizontal position. This configuration is shown in Figure 3.6.

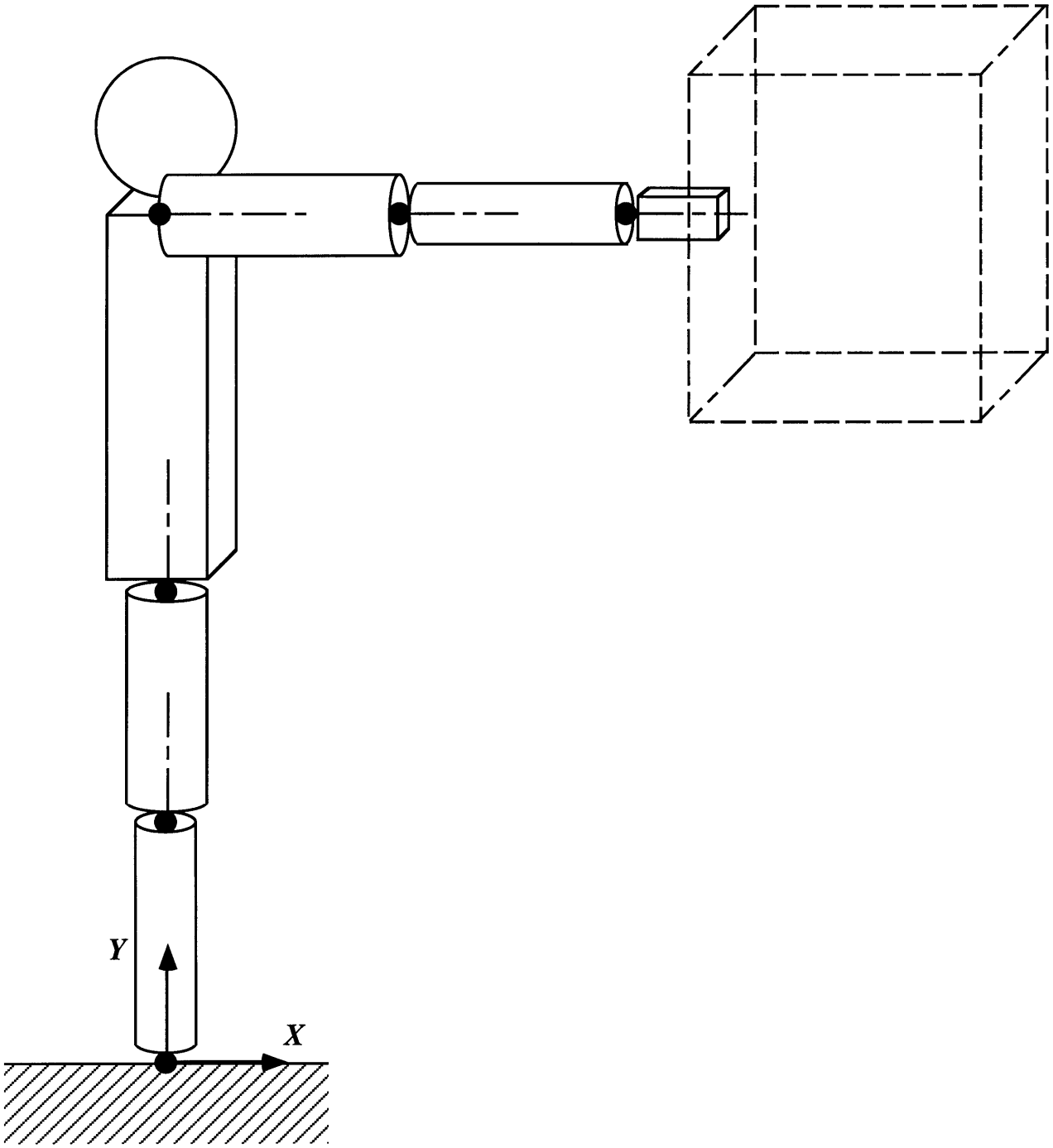


Figure 3.6 Initial configuration of system for application of test functions.

During each of the three phases, one of the joints is assigned a prescribed acceleration of 5 deg/sec^2 while all the other joints (including the hip, knee, and ankle) are assigned a prescribed acceleration of 0 deg/sec^2. In this way the bodies outboard of the test joint rotate as if they were one rigid body, while the bodies inboard of the test joint remain fixed because they are ultimately joined to ground at the ankle joint. For the outboard bodies moving in unison, it was possible to calculate their combined mass moment of inertia using the parallel axis theorem

$$I = \sum_{i=1}^{n} (I_{c.m.} + d^2 m)_{(i)} \tag{3.14}$$

where

I = total moment of inertia
$I_{c.m.}$ = moment of inertia of body i with respect to its center of mass
d = distance of body i center of mass from the joint under consideration
m = mass of body i
n = number of bodies outboard of joint under consideration

The torque applied at a joint can be calculated from the effective moment of inertia of the outboard bodies and the acceleration at the joint using the rotational (Euler) form of Newton's second law, $\tau = I\alpha$. A comparison of the computer calculated and hand calculated torque values for these test functions is presented in the next chapter.

3.4.4 Inverse Kinematics

The next step in the simulation is the inverse kinematics phase. It is termed *inverse* kinematics because only the motion of the endpoint is prescribed and this is used to determine the corresponding motion of the system in terms of joint coordinates. Before executing the motion, it is necessary to place the system in its initial configuration. It was chosen to place the trunk and legs in the neutral body posture for weightlesssness as specified in NASA Man Systems Standard 3000. The joint angles for an unsuited crewmember are used since no values for neutral body posture for a spacesuited crewmember could be found. The arms are placed in an initial configuration that allows for the Spartan 204 to be conveniently manipulated along a circular trajectory.

To simplify the simulation, it was specified that the ankle, knee, and hip joints should remain fixed at their starting angles. This was accomplished by prescribing zero angular acceleration and zero velocity for each of these joints. In addition, the hand is maintained

in a fixed orientation so that the Spartan payload does not rotate as the hand follows the prescribed trajectory. By simplifying the motion in this way, the redundant degrees of freedom are canceled by the constraints and only two remaining degrees of freedom, in the elbow joint and shoulder joint, remain to match the two degrees of freedom in the endpoint (hand) coordinates. This strategy is in keeping with the philosophy of getting a simpler simulation to work before modifying it to represent a more complex situation.

The simulation system is able to obtain solutions to inverse kinematics when redundant degrees of freedom are present by employing a linearized least squares root finder to determine the joint angles (and their derivatives) required to achieve the prescribed endpoint motion. If more degrees of freedom are allowed, however, then it becomes necessary to implement some form of control in some of the joints or unrealistic motion will result. More explicitly, if all the joints in the body are passive, and the hand is "dragged" along a certain trajectory, then the body will behave somewhat like a rag-doll where its posture is determined only by the mass properties of the segments. One way of controlling the posture of the body is to specify springs and dampers at certain joints which apply torques in proportion to the displacement and angular velocity of those joints. These springs and dampers mimic the passive behavior of muscle groups actuating a joint in the human body. The simulation can be modified slightly by including torsional springs and dampers in the ankle, knee, and hip joints. These joints then seek to achieve a certain prescribed angle but are allowed some play based on the compliance of the springs and dampers and this results in a somewhat more realistic motion than the rigid posture obtained by fixing the joint angles. This technique is implemented in a second simulation.

The inverse kinematic phase of the simulation now proceeds with the lower leg, upper leg, and trunk segments maintaining a fixed position while the upper arm, forearm, and hand move in appropriate ways to follow the circular trajectory while satisfying the constraints mentioned above. A circle was chosen as the trajectory because of its simplicity and because the smoothness reduced the peak accelerations (and thus peak forces and moments) required in the arm joints. To follow a square path at a constant speed, for instance, would require infinite accelerations (and infinite loads) in the joints at the corners. The initial configuration of the body and the circular trajectory followed by the hand are shown in Figure 3.7.

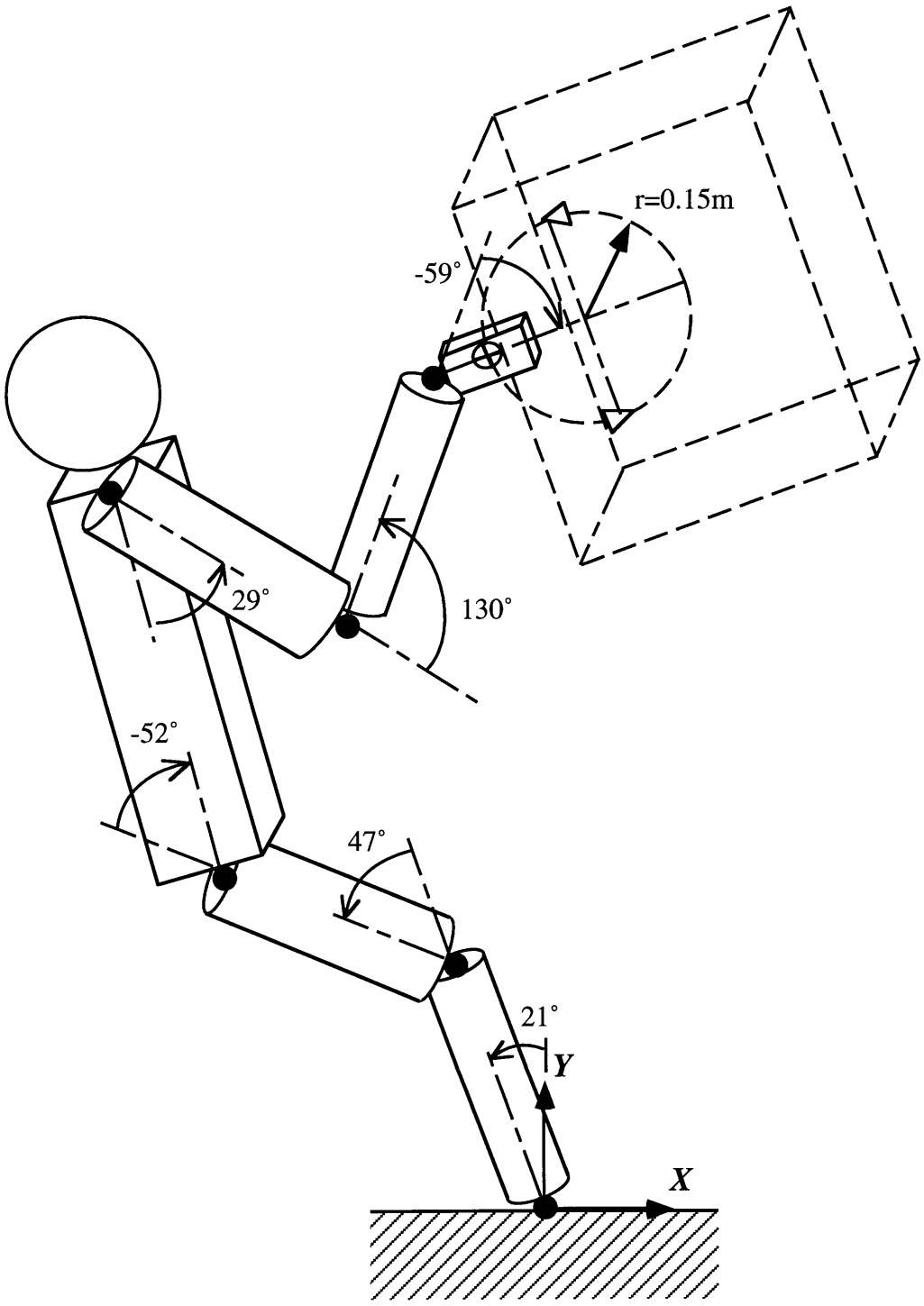


Figure 3.7 Initial configuration for first simulation and prescribed circular trajectory of hand.

The initial conditions are obtained by performing an assembly and initial velocity analysis through calls to SD/FAST routines. The outcome of these calls is a fully compatible state vector (the first half of the state vector represents all the position values in the system, and the second half represents all the velocity values.) Following this, the simulation code calls a routine which integrates the arm motion repetitively through small time steps (0.05 sec was used) until the full trajectory is completed. During each time step, the position, velocity and acceleration of the center of mass of the hand were prescribed using the following expressions:

$$\begin{aligned}
&\theta = \pi + \omega t \\
&\begin{cases} x_{cm} = x_{start} + r(1+\cos\theta) \\ y_{cm} = y_{start} + r\sin\theta \end{cases} \\
&\begin{cases} \dot{x}_{cm} = -\omega r\sin\theta \\ \dot{y}_{cm} = \omega r\cos\theta \end{cases} \\
&\begin{cases} \ddot{x}_{cm} = -\omega^2 r\cos\theta \\ \ddot{y}_{cm} = -\omega^2 r\sin\theta \end{cases}
\end{aligned} \tag{3.15}$$

where,

ω = angular velocity
r = radius of circular trajectory

At each incremental time step during the motion simulation, the values for position, velocity, and acceleration of each joint in the body (excluding the sliding joints connecting the massless bodies) are recorded in a two dimensional state-time array. These values are subsequently recalled in a "playback" mode during the inverse dynamics phase described next.

3.4.5 Inverse Dynamics

During the inverse dynamics phase of the simulation, the values of joint position, velocity, and acceleration determined during the inverse dynamics phase are recalled and used to prescribe the motion of the system in a "playback" mode. Prescribed motion is turned "on" for the pin joints and "off" for the sliding joints and the one pin joint connecting a massless body to the hand. After each time step, the joints are assigned the prescribed values associated with the point in time during the motion and the simulation calls an SD/FAST function, called "sdhinget", for each joint to determine what torque is

required in that joint to achieve the required motion. The resulting time histories for torque are presented in the next chapter.

3.5 Comparison with Physiological Limits

Once the data for the position, velocity, acceleration, and torque of the joints has been obtained, it can be further analyzed to evaluate the physiological state of the crewmember in the simulation. It is of interest to compare the values obtained with human physiological limits such as joint range of motion and muscle strength.

Due to the preliminary nature of this simulation and the philosophy of beginning with a simple case, there is as yet no mechanism built into the simulation which enforces the physiological limits of human motion. The joint position, velocity, acceleration, and torque values calculated are entirely theoretical and represent the values required to accomplish the manipulation task while obeying the laws of multibody dynamics. There is no guarantee that these values fall within the range of human capability. For this reason, it is very useful to compare these theoretical values with experimentally determined values of human performance. Of particular interest are the limits of joint range of motion and of maximum joint actuation strength. Joint range of motion limits are obtained from the "NASA Man-Systems Integration Standards" (NASA 1987) and are plotted along with the theoretical values of joint angle. Joint actuation strength is represented by torque values as a function of joint angle and joint velocity. These maximum torque values are obtained from a human strength model developed by the Anthropometrics and Biomechanics Laboratory at NASA's Johnson Space Center (Pandya, Hasson et al. 1992; Pandya, Maida et al. 1992).

3.6 Animation and Data Display

An animation and data display program called EVADS (EVA Dynamic Simulation) has been developed for the purpose of communicating graphically both qualitative and quantitative information about the simulation. Both animation and data plots are displayed simultaneously so that the plots can be correlated with the three dimensional image of the system.

Data for this program is input in the form of text files created by the simulation driver code. Due to the large quantity of data, especially in systems that have several segments, four separate files are input for position, velocity, acceleration, and torque data. In the future it is hoped that this program can be linked with the simulation driver code so that the simulations can be controlled in a more visually interactive manner.

The three-dimensional rendered images in the animation are of great help to the analyst. One can determine at a glance whether the overall system starts off with the correct initial configuration and executes motion that makes sense. The user can very conveniently alter the viewing angle and size of the animation image by means of the mouse controls and simple keystrokes.

The data is plotted below the animation portion of the window. At present, only one data category (e.g., elbow torque) can be plotted at a time, but there are plans to allow multiple plots to be displayed simultaneously in the future. The parameter to be displayed is selected from an array of virtual buttons arranged vertically along the right side of the screen. The plot itself is displayed in green with a fine grid in the background. The vertical axis displays the parameter chosen and the scale is automatically chosen to accommodate the limiting values in the data set. The horizontal axis always displays time and the scale is fixed. To allow for time histories of varying length, however, the horizontal axis has the ability to scroll left and right. The point in time representing the state of the animated image is identified on the plot by means of a red vertical line. During the course of an animation run, this line moves along the plot in synchronization with the animated image. It is also possible to step through the animation manually, either forwards or backwards, using the ">" and "<" keys respectively. This mode is particularly useful for observing various parameters at specific points in time during the simulation and for debugging. Figures showing the computer images generated by EVADS are presented in the next chapter.

4. Results

This chapter presents the results of two dynamic simulations. The multibody model and associated system description file is the same for both simulations. Equations of motion are formulated by SD/FAST (using Kane's method) and represented in an implicit computational form. Specific simulation code performs four analyses on the dynamic system: joint torque test functions, assembly and initial velocity analysis, inverse kinematics, and inverse dynamics. The data is then visualized on the computer by means of three dimensional rendered animation and parameter time history plots.

The following sections present the results of: a comparison between the computer calculated and hand calculated joint torques for the test functions; the numerical results of the first simulation; the numerical results of the second simulation; and the animation and parameter plots of the EVADS interface. Numerical results for each of the two simulations are described in two subsections: data obtained from the inverse kinematics phase (joint angles, velocities, and accelerations) and joint torques obtained from the inverse dynamics phase.

4.1 Joint Torque Test Functions

The correlation between the hand calculated and computer calculated torque values for the test conditions described in the previous chapter are shown in Table 4.1. Considering that the parallel axis theorem was used to approximate the moments of inertia of segments moving in unison, the correspondence between values is remarkably good. The percent error was calculated by subtracting the torque values from the computer simulation from the hand calculated torque value, dividing the difference by the former, and multiplying by 100. The largest error, a value of -0.030 %, occurs in the wrist joint. While this value is still very low, the slightly larger error is probably accounted for by the close proximity of the wrist joint to the center of mass of the Spartan payload, reducing the accuracy of the parallel axis approximation.

Table 4.1 Comparison of hand calculated and computer simulation torques for test conditions in shoulder, elbow, and wrist joint.

Joint	Torque from Hand Calculation [N-m]	Torque from Computer Simulation [N-m]	Percent Error
Shoulder	211.441	211.419	+0.010
Elbow	138.196	138.210	-0.010
Wrist	86.951	86.977	-0.030

4.2 Simulation No. 1 - Fixed Lower Body

Two simulations of an EVA task were run. In the first simulation, the lower body joints (ankle, knee, and hip) were held in fixed positions, while in the second simulation, the lower body joints were given some compliance by means of virtual springs and dampers representing the passive mechanical properties of these joints. The numerical results of each simulation run are presented in a series of four figures. The first three categories, namely, joint angle, joint velocity, and joint acceleration are presented in this subsection for the fixed lower body simulation. Joint torques are presented in the following subsection. Each of the figures consists of a composite of six subplots depicting the data obtained for the ankle, knee, hip, shoulder, elbow, and wrist joints. In each case the relevant parameter is plotted against time as the independent variable. In all cases the curve depicting the actual kinematic or dynamic state of the system is shown as a solid line. Where limits on the parameter are available, for instance joint range of motion or maximum torque, the upper limit is plotted as a dashed line while the lower limit is plotted as a dash-dot line. For convenience, this key is summarized in Table 4.2 below. In viewing the plots it should be noticed that the vertical axis varies according to the range of data values represented. The automatic scaling of the plots helps to bring out details, but should be carefully considered when comparing plots with one another. The complete set of numerical data used to create the plots of the various system parameters and their limits is available in a separate document held at the MIT Man-Vehicle Laboratory. For conciseness, this chapter lists values only for specific conditions, such as the maxima and minima of each relevant parameter.

Table 4.2 Key for interpreting multi-curve plots.

Curve	Line Type
Upper Limit	--------
Actual State of System	_____
Lower Limit	-.-.-.-.-.

Before looking at the numerical data in depth, it is helpful to obtain a mental picture of the simulation run. For this reason, a composite of six images is shown in Figure 4.1. Each of these images shows the state of the system at intervals of 2.0 seconds, beginning with the initial configuration and ending with the final configuration. In this case, the initial and final configurations are exactly the same since the motion involves manipulating the object through a circular trajectory.

It can clearly be seen how the lower body (lower leg, upper leg, and trunk) is stationary, while the motion is carried out by the arm segments alone. Notice also that the Spartan payload is kept at a constant orientation throughout the simulation and experiences only translational motions. The next subsection presents the results of the inverse kinematics analysis in the first simulation, followed by a subsection presenting the joint torques found in the inverse dynamics analysis.

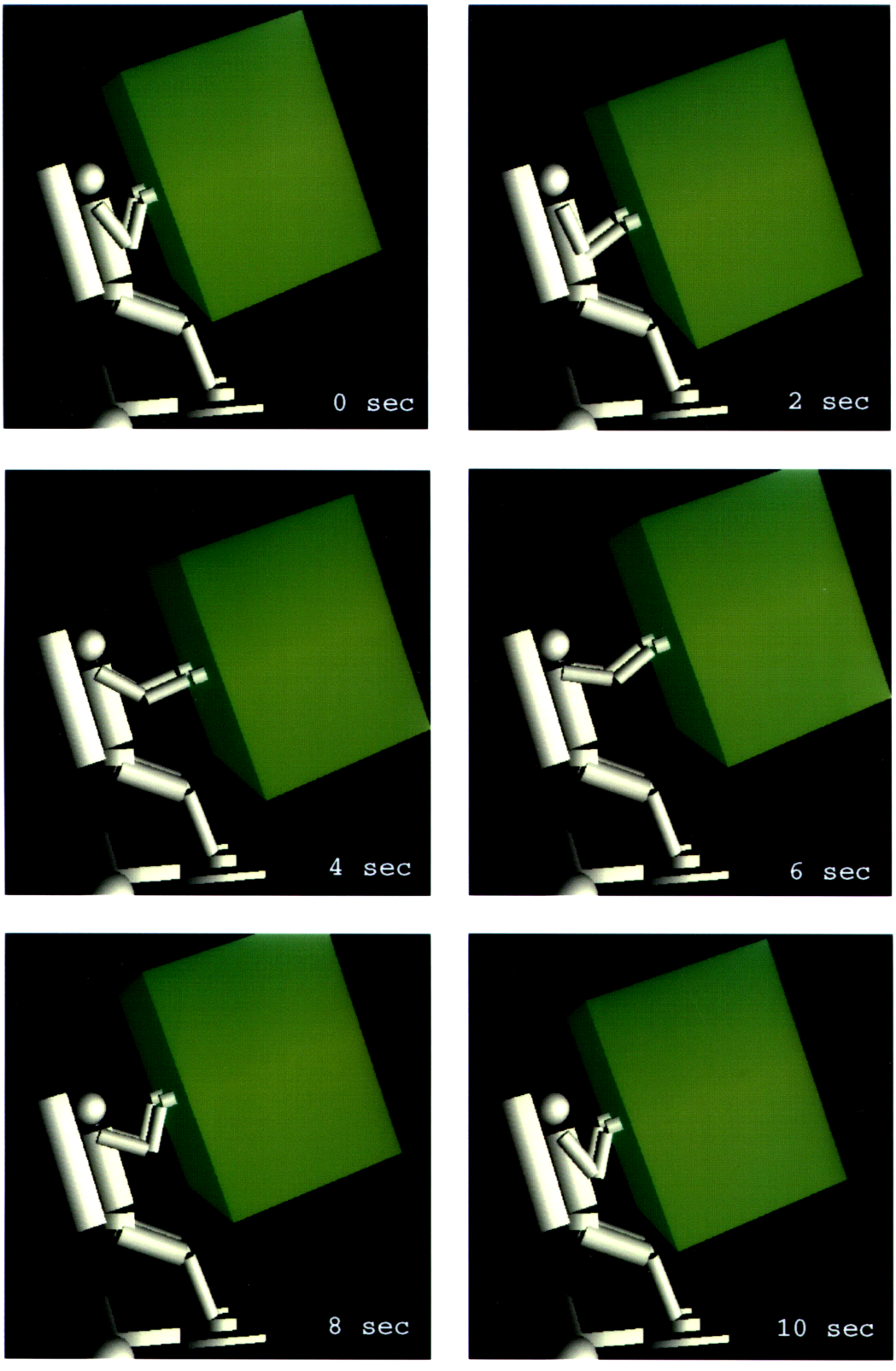


Figure 4.1 Animation sequence for simulation no. 1 – fixed lower body. Intervals are 2 seconds.

4.2.1 Inverse Kinematics (No. 1 - Fixed Lower Body)

The first category of data, namely, joint angles, is presented in Figure 4.2. The limits on joint range of motion were obtained from the listings for the EMU (Shuttle spacesuit) mobility in the NASA "Man-Systems Integration Standards" (NASA-STD-3000) publication (NASA 1987). The range of motion values are summarized in Table 4.3. In general, the values were calculated by taking the average between the 5th and 95th percentile values which were originally derived from measurements on a statistically large population of test subjects. The exception to this method was the wrist joint, for which spacesuit data was not available for the particular joint axis desired (wrist radial and ulnar deviation). Wrist values were calculated by taking 85% of the unsuited (shirt sleeve environment) values listed in the same reference. The fraction of 85% was used because this is the general estimated mobility for the EMU, which was also obtained from NASA-STD-3000.

Table 4.3 EMU joint range of motion limits (edited from NASA-STD-3000).

Joint	Lower Limit [deg]	Upper Limit [deg]
Ankle	-40.0	40.0
Knee	0.0	120.0
Hip	-70.0	0.0
Shoulder	0.0	180.0
Elbow	0.0	130.0
Wrist	-28.3	22.8

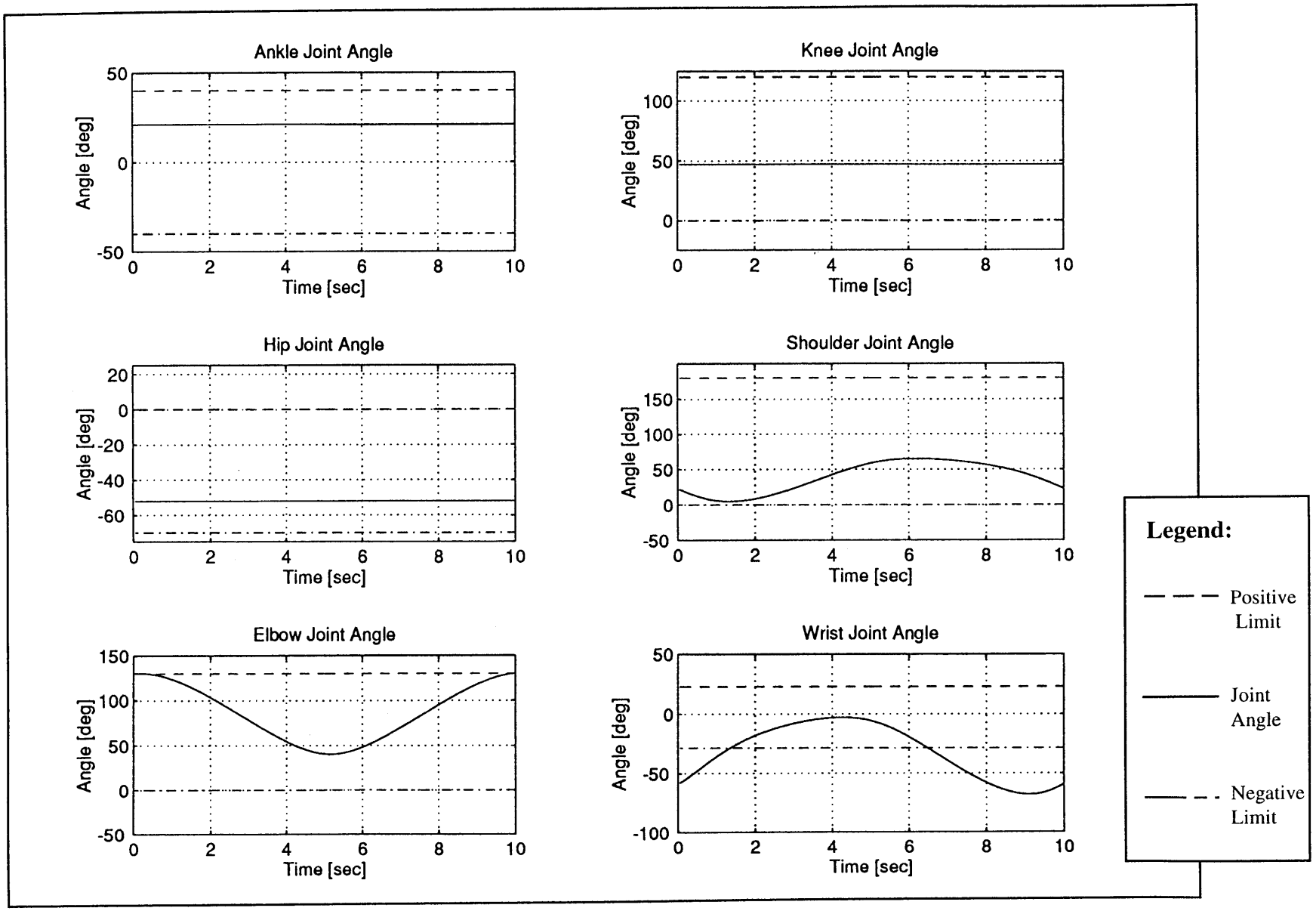


Figure 4.2 Joint angle plots for simulation no. 1 (fixed lower body).

As expected, the first three plots in Figure 4.2 confirm that the joint angles of the ankle, knee, and hip, remain fixed at the initial values of 21°, 47°, and -52° respectively. This is exactly what was intended since these joints were prescribed to remain at these angles throughout this particular simulation run.

The shoulder, elbow, and wrist plots are more interesting. The shoulder joint data follows a roughly sinusoidal trend. It starts at and returns to a value of 29°. Again, this is as expected since the hand c.m. describes a closed circle. The maximum and minimum angles reached in the three articulated joints are summarized in Table 4.4. Note that the term "minimum" refers to the lower of the two values on a signed scale so that a negative angle is called a "minimum" even if its absolute magnitude is larger than a "maximum" positive value. This convention is also followed for the velocity, acceleration, and torque parameters.

Table 4.4 Maximum and minimum angles reached by articulated joints during simulation No. 1 - Fixed Lower Body.

Joint	Time of Max. [sec]	Max. Angle [deg]	Time of Min. [sec]	Min. Angle [deg]
Shoulder	6.20	65.1	1.35	4.9
Elbow	0.00 & 10.00	130.0	5.15	40.4
Wrist	4.25	-2.9	9.10	-67.9

By observing the limiting curves for joint range of motion, it can easily be seen that the wrist is the only joint exceeding its range of motion. In this case, however, the excursions below the lower limit of -28.3° are so severe, the maximum deviation is -49.8°, the task becomes completely impossible. Changes incorporated in the second simulation sought to avoid this problem.

Values of joint velocity are displayed in Figure 4.3. As expected the ankle, knee, and hip joints exhibit zero velocity since their angles are fixed. The range of values attained by the remaining three joints are shown in Table 4.5. No limits are shown on the plots in Figure 4.3 because no data has been found on the limits of human joint velocity in a spacesuit.

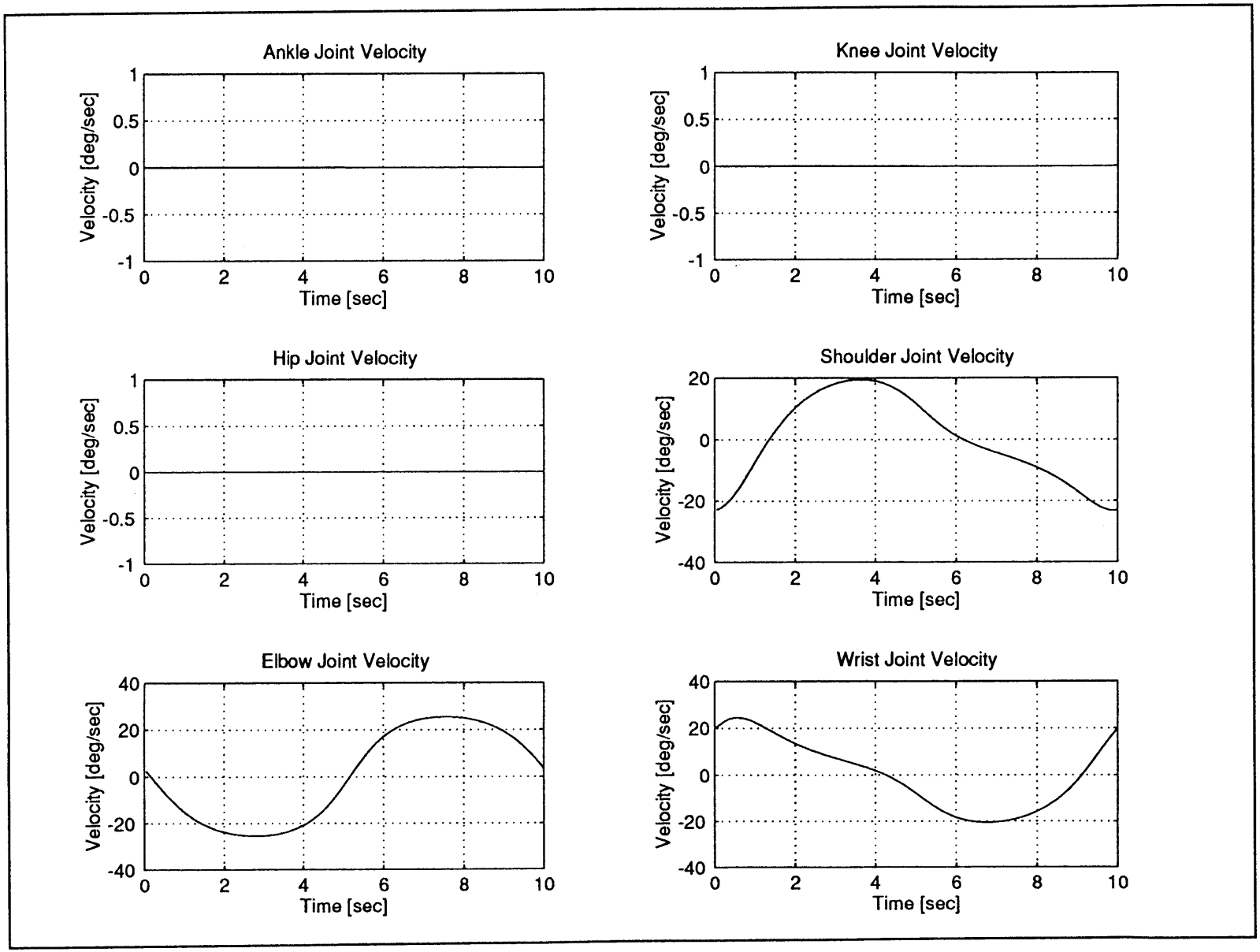


Figure 4.3 Joint velocity plots for simulation no. 1 (fixed lower body).

Table 4.5 Maximum and minimum velocities for articulated joints during simulation No. 1 - Fixed Lower Body.

Joint	Time of Max. [sec]	Max. Velocity [deg/sec]	Time of Min. [sec]	Min. Velocity [deg/sec]
Shoulder	3.65	19.4	9.90	-23.2
Elbow	2.80	-25.4	7.55	25.4
Wrist	0.55	24.5	6.75	-20.5

A quick check of the velocity data is available by comparing the velocity curves with the joint angle curves. It is observed that the joint velocity curves do in fact represent the derivative of the joint angle curves as expected.

The last data set that falls under the description of inverse kinematics, are the joint acceleration curves shown in Figure 4.4. Once again the ankle, knee and hip joints are fixed. The shoulder, elbow, and wrist joint acceleration curves are the first derivatives of the corresponding velocity curves and attain the extreme values shown in Table 4.6. Again, no data has been obtained on the limits of joint acceleration for a person wearing a spacesuit. In the case of acceleration, however, it would seem that these limits would be of little value, particularly since the mechanical properties of a spacesuit appear to depend primarily on joint angle (torsional spring force), with possibly some lower level forces from damping (velocity dependence). At any rate, since joint acceleration is related to joint torque, the limits on human strength, displayed along with the joint torque values, provides an indication as to whether acceleration values are realistic.

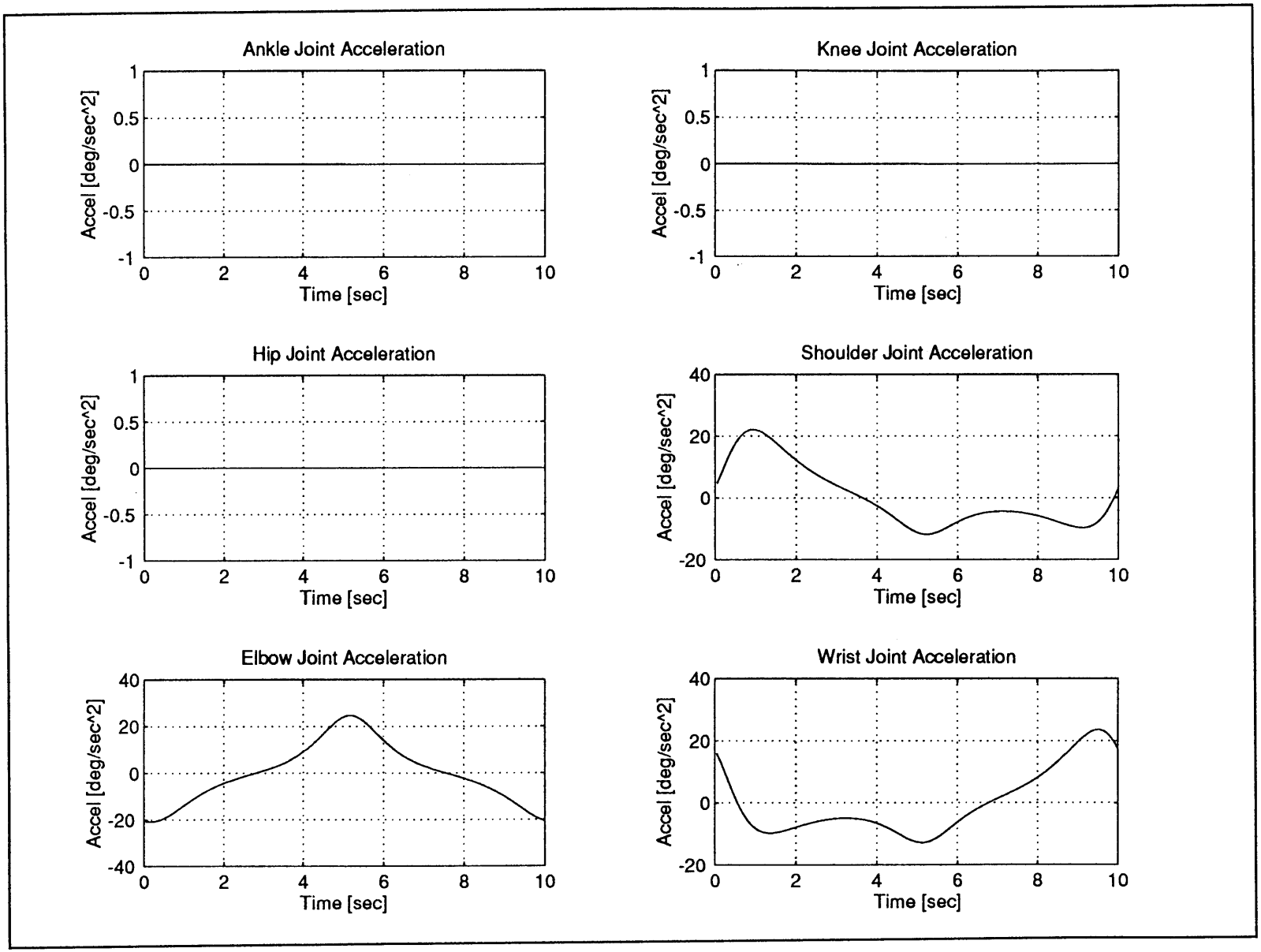


Figure 4.4 Joint acceleration plots for simulation no. 1 (fixed lower body).

Table 4.6 Maximum and minimum accelerations for articulated joints during simulation No. 1 - Fixed Lower Body.

Joint	Time of Max. [sec]	Max. Acceleration [deg/sec^2]	Time of Min. [sec]	Min. Acceleration [deg/sec^2]
Shoulder	0.95	22.1	5.20	-11.9
Elbow	5.15	24.6	0.15	-20.9
Wrist	9.50	23.6	5.10	-12.8

4.2.2 Inverse Dynamics (No. 1 - Fixed Lower Body)

Values of joint torque, obtained from the inverse dynamics phase of the first simulation, are displayed in Figure 4.5. In this case, the curves of the system joint torques reveal fairly smooth sinusoidal shapes. The biggest difference between this family of plots and those in the inverse kinematics section is the fact that the ankle, knee, and hip joints now exhibit non-zero and non-stationary values. Non-zero torques should be expected because even though the joints maintain a constant position, they must exert varying amounts of torque to hold that position as the configuration and torques in the rest of the system change. One might think of these torques as reaction torques. The maximum and minimum values reached by the system torque curves are listed in Table 4.7.

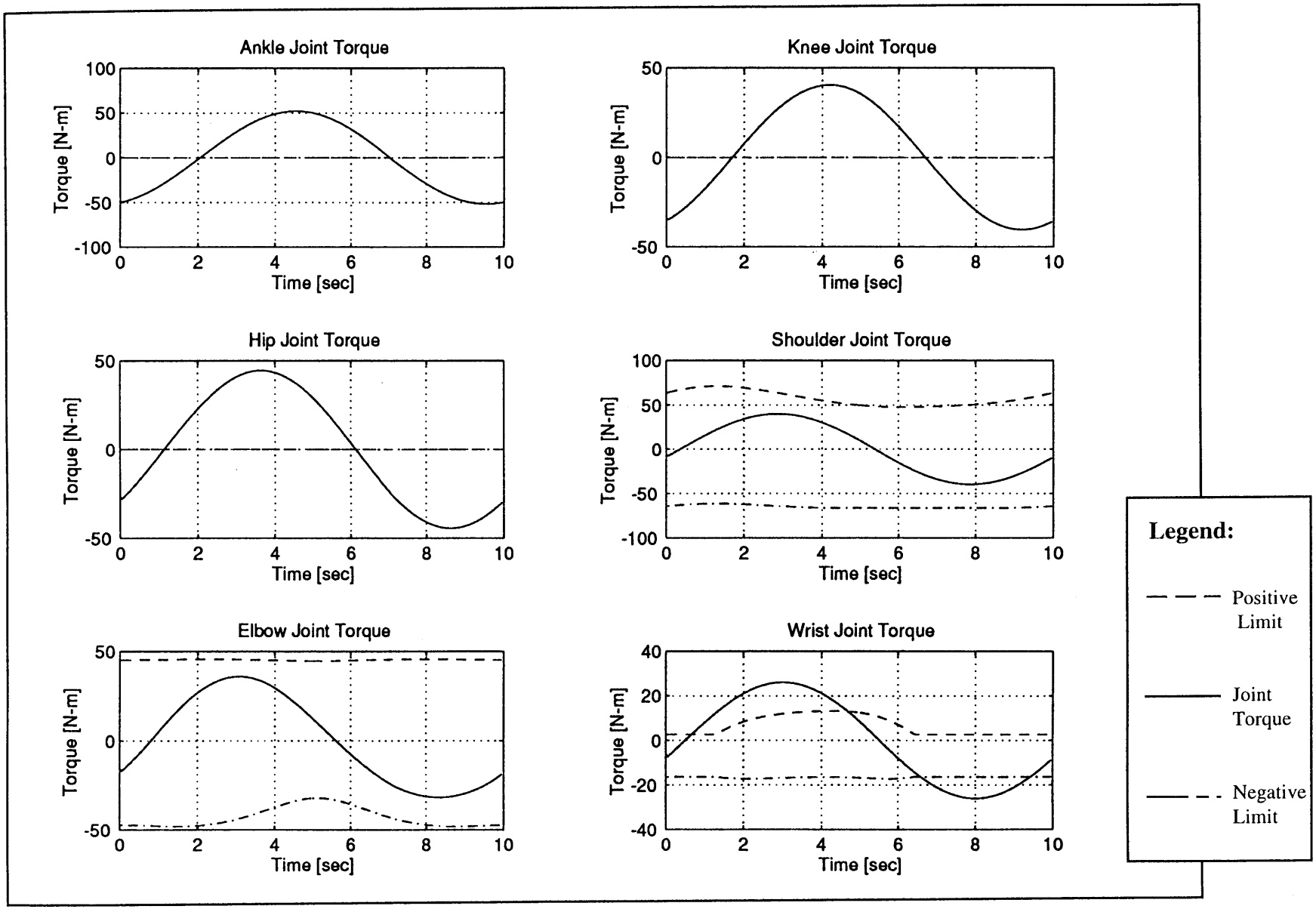


Figure 4.5 Joint torque plots for simulation no. 1 (fixed lower body).

Table 4.7 Maximum and minimum torques reached by system joints during simulation No. 1 - Fixed Lower Body.

Joint	Time of Max. [sec]	Max. Torque [N-m]	Time of Min. [sec]	Min. Torque [N-m]
Ankle	4.55	52.0	9.55	-51.8
Knee	4.20	40.5	9.20	-40.3
Hip	3.65	44.6	8.65	-44.5
Shoulder	2.85	39.9	7.85	-39.9
Elbow	3.05	36.0	8.30	-31.8
Wrist	3.00	26.1	8.00	-26.1

Three interesting facts surface upon surveying the values in this table. Firstly, all of the joints with the exception of the elbow joint, exhibit maxima and minima that are exactly 5.00 seconds apart (half of the total simulation time). The time between the maximum and minimum torque values for the elbow joint is a slightly larger value of 5.25 seconds. Secondly, all of the joints with the exception of the elbow joint, have a maximum and minimum torque value that are symmetrical around zero Newton-meters. The elbow joint average torque is slightly offset from zero at a value of 4.25 N-m. Thirdly, it is clear that the ankle joint experiences the highest torque values in both positive and negative senses, which is not too surprising since the ankle joint is related to the object being manipulated, and thus to the endpoint force exerted, by the longest moment arm in the body. Furthermore, the ankle is not assisted in providing reaction torques by the weight of the astronaut's body, as would be the case in a one-g environment.

4.3 Simulation No. 2 - Compliant Lower Body

The second simulation is an improvement on the first. Two objectives were followed during the creation of this simulation. Firstly, violations of human physiological constraints such as joint range of motion and torque were avoided, and secondly, additional features were incorporated into the simulation to make the results more realistic than those obtained in the first simulation.

The most significant concern that came out of an observation of the results from the first simulation was the deviation of the wrist joint from its allowable range by a very large amount. To avoid this problem, the initial configuration of the astronaut was altered slightly. The ankle, knee, and hip joint were set to the same neutral body posture angles

as before (21°, 47°, and -52°). This time, however, the shoulder joint was set to an initial angle of 0°; the elbow joint was set to an initial angle of 90°; and the wrist joint was set to an initial angle of 0°. This means that the wrist starts off in a comfortable position close to the center of its range of motion. The initial configuration for this simulation can be seen in Figure 4.12.

To reduce the large joint torques observed in the first simulation, the angular velocity with which the hand c.m. traces out the circular trajectory was reduced to half of the value of the first simulation so that the circle is completed in 20 seconds instead of 10 seconds. Initially, the simulation was attempted with the same radius as before (0.15 m). Unfortunately, due to the new starting position, the arm ran into a singular configuration (full extension) because part of the circle lay outside of the reach envelope of the c.m. of the hand. To avoid this condition, the radius of the circular trajectory was reduced to 0.75 m. This also helped to reduce the required joint torques as explained below.

Instead of fixing the ankle, knee, and hip joints, torsional springs and dampers were added to these joints to provide some passive compliance. It is a reasonable approximation to model the passive impedance of these joints in this way since it is well known that muscle actuators exhibit the gross mechanical properties of both elasticity and damping (McMahon 1984). In a way, these springs and dampers act like proportional-plus-derivative controllers which try to maintain their respective joints at the desired angles by exerting torque proportional to the angular deviation of the joint from the desired angle (spring) and proportional to the angular velocity of the joint (damper) according to the relation

$$\tau_{\text{joint}} = -k_{rot}\left(q_{\text{joint}} - q_{\text{bias}}\right) - b_{\text{damp}}\dot{q}_{\text{joint}} \tag{4.1}$$

where

τ_{joint} = passive torque exerted on joint
k_{rot} = spring constant in N - m / deg
q_{joint} = joint angle measured from reference position
q_{bias} = joint bias angle (desired position)
b_{damp} = damping constant
$\dot{q}_{\text{joint}}$ = joint angular velocity

An equation of this form is applied to each of the ankle, knee, and hip joints. Based on estimates of the order of magnitude of human joint torque strength, k_{rot} was chosen to be

100 N-m/rad and b_{damp} was chosen to be 10 N-m/(rad/sec) for all three joints. In addition, each of the three joints are provided with very stiff "joint stop" springs (k_{rot} = 1000 N-m/rad) that are activated if the joint angle exceeds its limits.

The hand is prescribed to remain at the orientation of its initial configuration (w.r.t. ground) so that the orientation of the Spartan payload remains fixed and it executes only translational motions. The elbow and shoulder are allowed complete freedom in following the prescribed endpoint trajectory, as in the first simulation. The logic behind the choice of conditions described above, including the placement of springs and dampers, is that the astronaut uses his lower body in a passive way, in an attempt to maintain a certain posture, while he uses his arms to move the payload along the desired trajectory. The wrist joint is used to keep the payload at a fixed orientation.

As with the first simulation, the sequence of motions visualized in the animation of simulation no. 2 are shown in the six image composite of Figure 4.6. The most noticeable difference between the animation of this simulation and the animation of the first simulation is that the astronaut's body sways back and forth as the object is being manipulated. At first the astronaut's body tilts backward, mainly due to ankle extension, in reaction to the force he exerts on the payload as he pushes it away from his body. As the payload reaches the furthest point from his chest (halfway point in trajectory), his body tilts forward, mainly due to ankle flexion, in reaction to the force with which he pulls the Spartan payload toward his body.

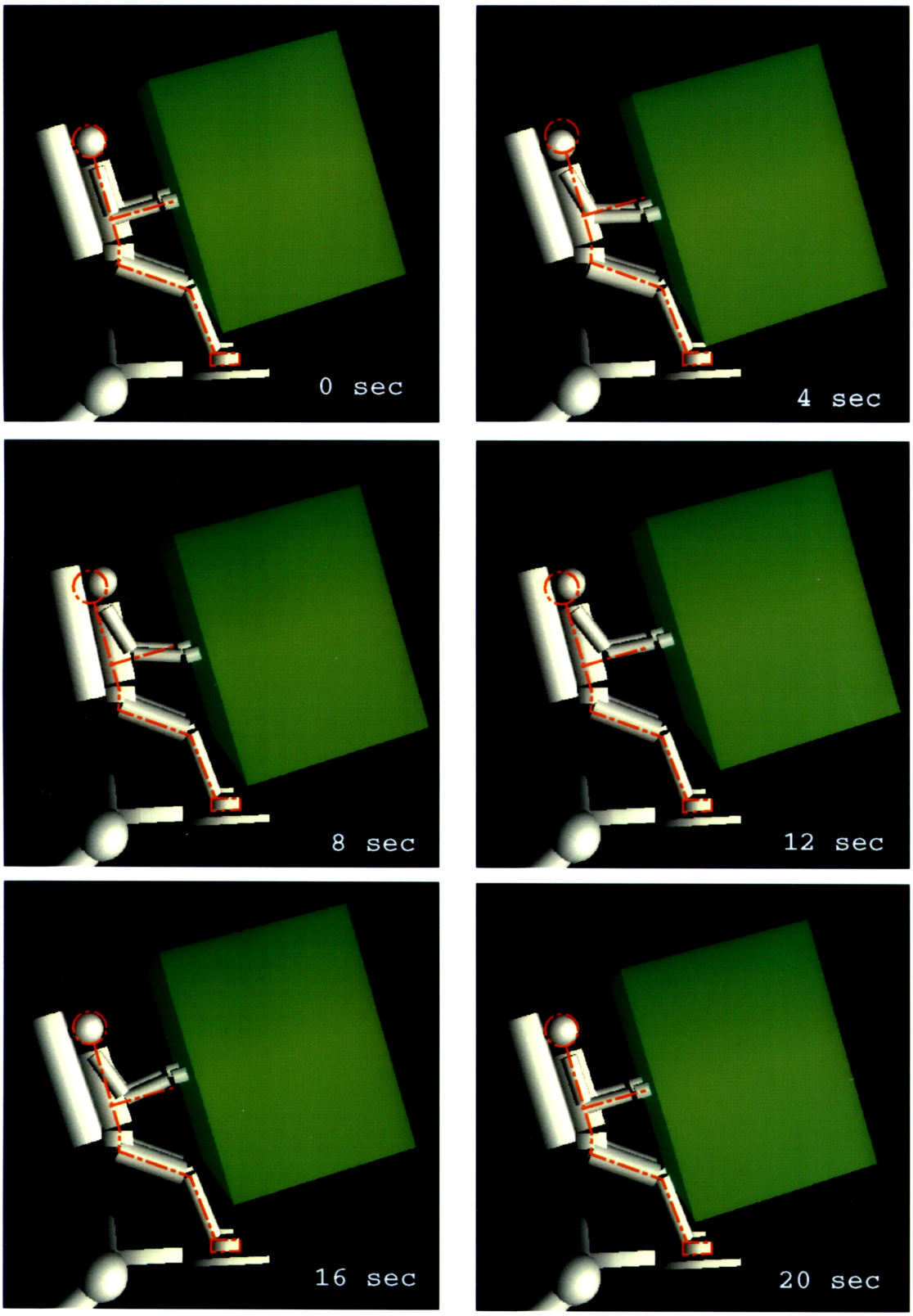


Figure 4.6 Animation sequence for simulation no. 2 – compliant lower body. Intervals are 4 seconds. Red dashed lines indicate starting configuration of body to accentuate lower body motions..

4.3.1 Inverse Kinematics (No. 2 - Compliant Lower Body)

Values of joint angle are revealed in the plots shown in Figure 4.7. Maxima and minima for each joint angle are summarized in Table 4.8. The first noticeable difference between this simulation and the previous one is that the lower body joints (ankle, knee, and hip) are no longer stationary. Instead, they execute small oscillations around their starting angles. An important difference in the second simulation is that the wrist angle time history is entirely contained within the range of motion limits. All of the other joint angles are contained within their range of motion limits, except for a slight dip below the lower limit of the shoulder joint. This violation could easily be remedied by either starting the shoulder joint at a slightly positive initial angle or by incorporating stiff spring joint-stops in the shoulder.

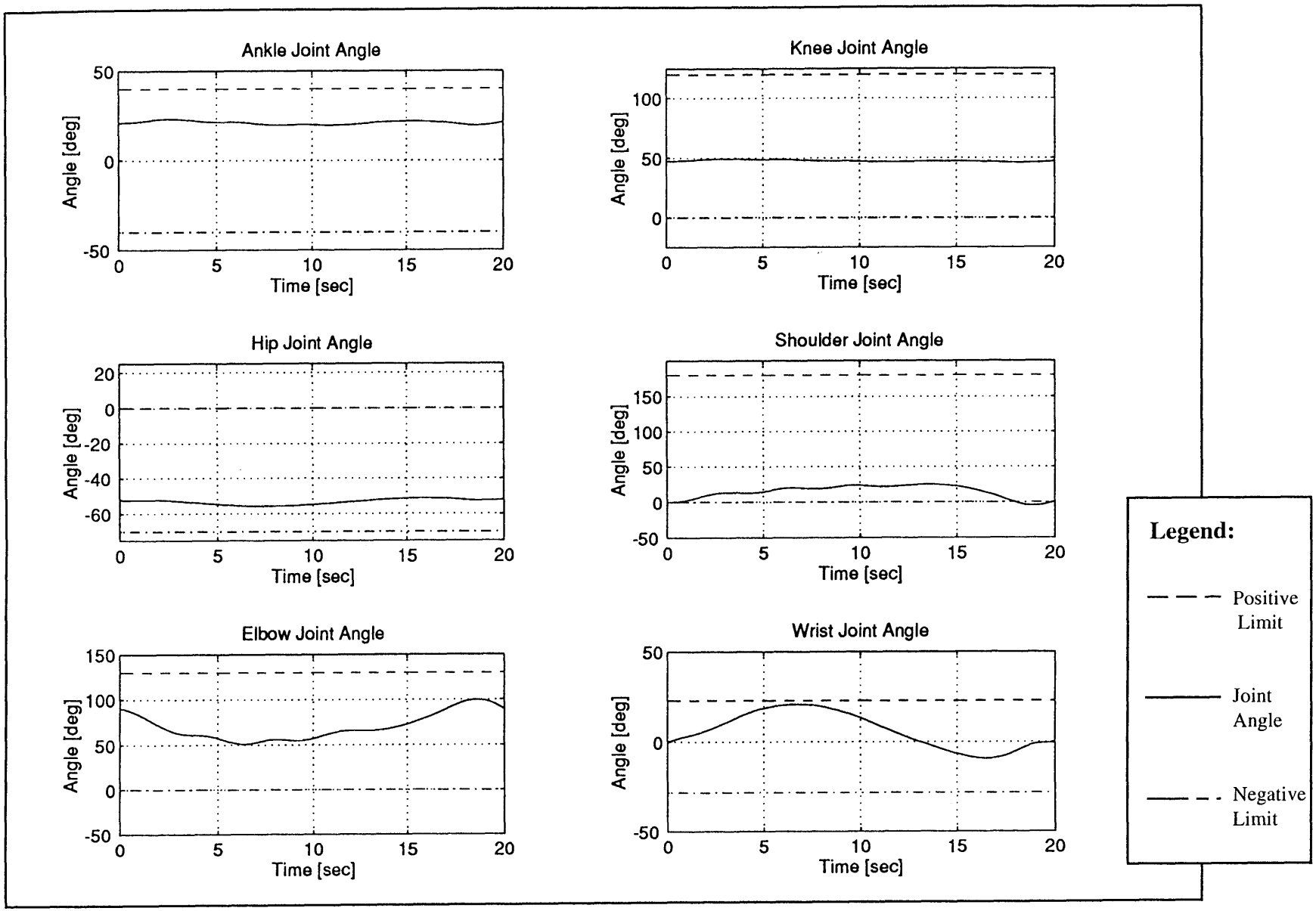


Figure 4.7 Joint angle plots for simulation no. 2 (compliant lower body).

Table 4.8 Maximum and minimum angles reached by articulated joints during simulation No. 2 - Compliant Lower Body.

Joint	Time of Max. [sec]	Max. Angle [deg]	Time of Min. [sec]	Min. Angle [deg]
Ankle	2.75	23.13	18.65	19.43
Knee	3.05	49.23	18.45	46.04
Hip	15.95	-51.14	7.15	-55.69
Shoulder	13.65	25.06	18.95	-4.82
Elbow	18.55	99.66	6.45	50.92
Wrist	6.85	20.72	16.55	-9.26

Angular velocity values for the various joints are plotted in Figure 4.8. The summary of maxima and minima for joint velocities is given in Table 4.9. The velocity values for the ankle, knee, and hip exhibit small fluctuations around an approximate mean of 0 deg/sec. The initial oscillations that last for the first half of the simulation period are accounted for by the sudden start of the manipulation task causing abrupt peaks in acceleration at the beginning of the simulation (seen in the acceleration plots of Figure 4.9). The velocity oscillations gradually die down due to the slight damping in the lower body joints.

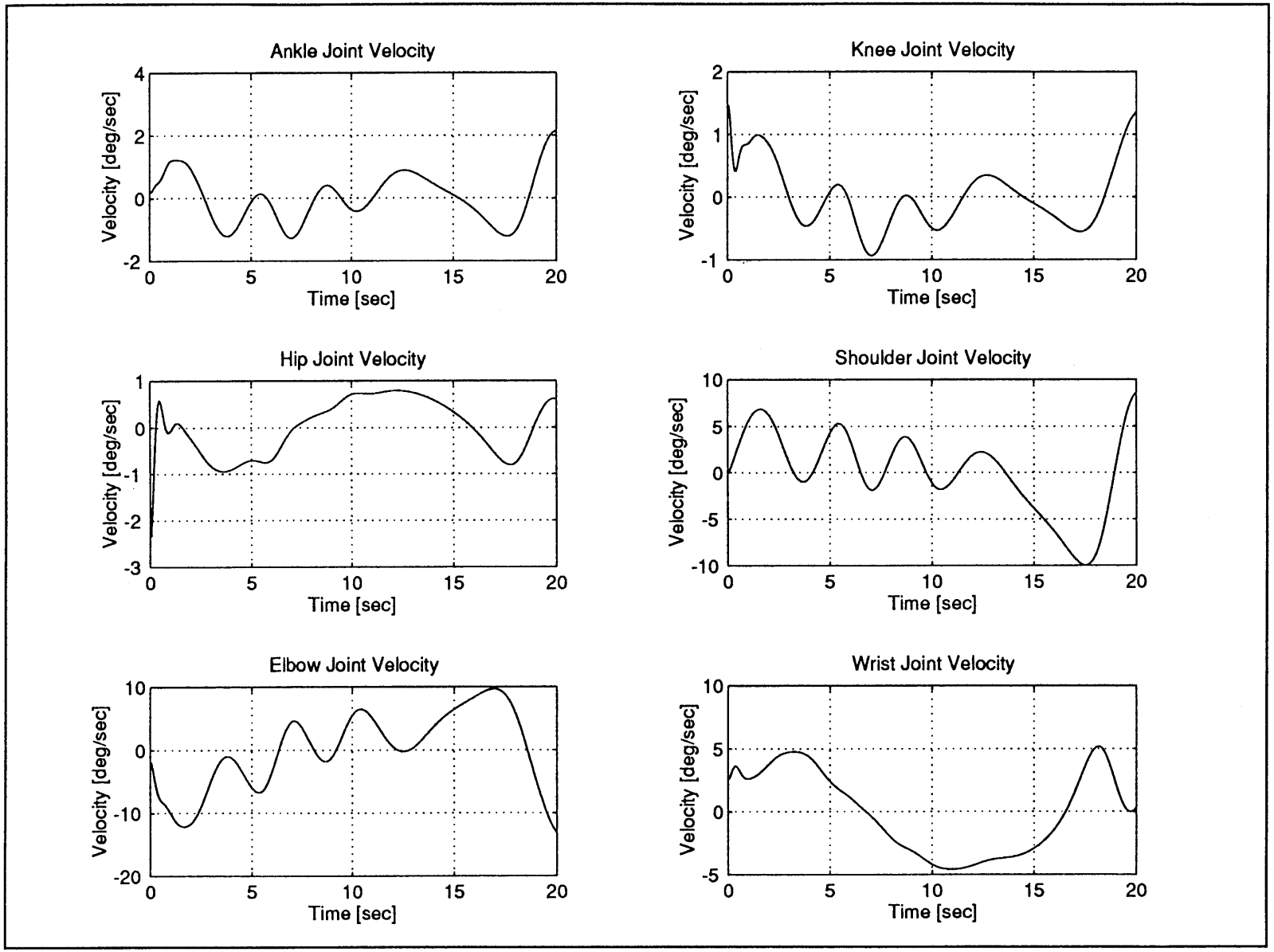


Figure 4.8 Joint velocity plots for simulation no. 2 (compliant lower body).

Table 4.9 Maximum and minimum velocities for joints during simulation No. 2 - Compliant Lower Body.

Joint	Time of Max. [sec]	Max. Velocity [deg/sec]	Time of Min. [sec]	Min. Velocity [deg/sec]
Ankle	20.00	2.29	7.05	-1.15
Knee	20.00	1.15	7.00	-1.15
Hip	12.30	0.57	0.10	-1.72
Shoulder	20.00	8.59	17.50	-9.74
Elbow	17.00	9.74	20.00	-13.18
Wrist	18.10	5.16	11.00	-4.58

The acceleration plots, displayed in Figure 4.9, show that the lower body joints experience slight acceleration, although usually on a smaller scale than the arm joints. All of the joint accelerations fluctuate around a zero mean. Most of them also exhibit a sharp acceleration spike at the beginning of the simulation run, as mentioned above, caused by the discontinuity in velocity at the beginning of the manipulation task. These spikes cause oscillations, but they appear to dissipate by the time the simulation is into the second half of its run time due to the damping in the lower body joints. The joint torque curves for the arm joints are much smoother (almost sinusoidal) than the lower body joints due to the smoothing effect of the linearized least squares solver applied to these joints during the inverse kinematics phase and because these joints are not controlled by springs and dampers. The maximum and minimum acceleration values are summarized in Table 4.10.

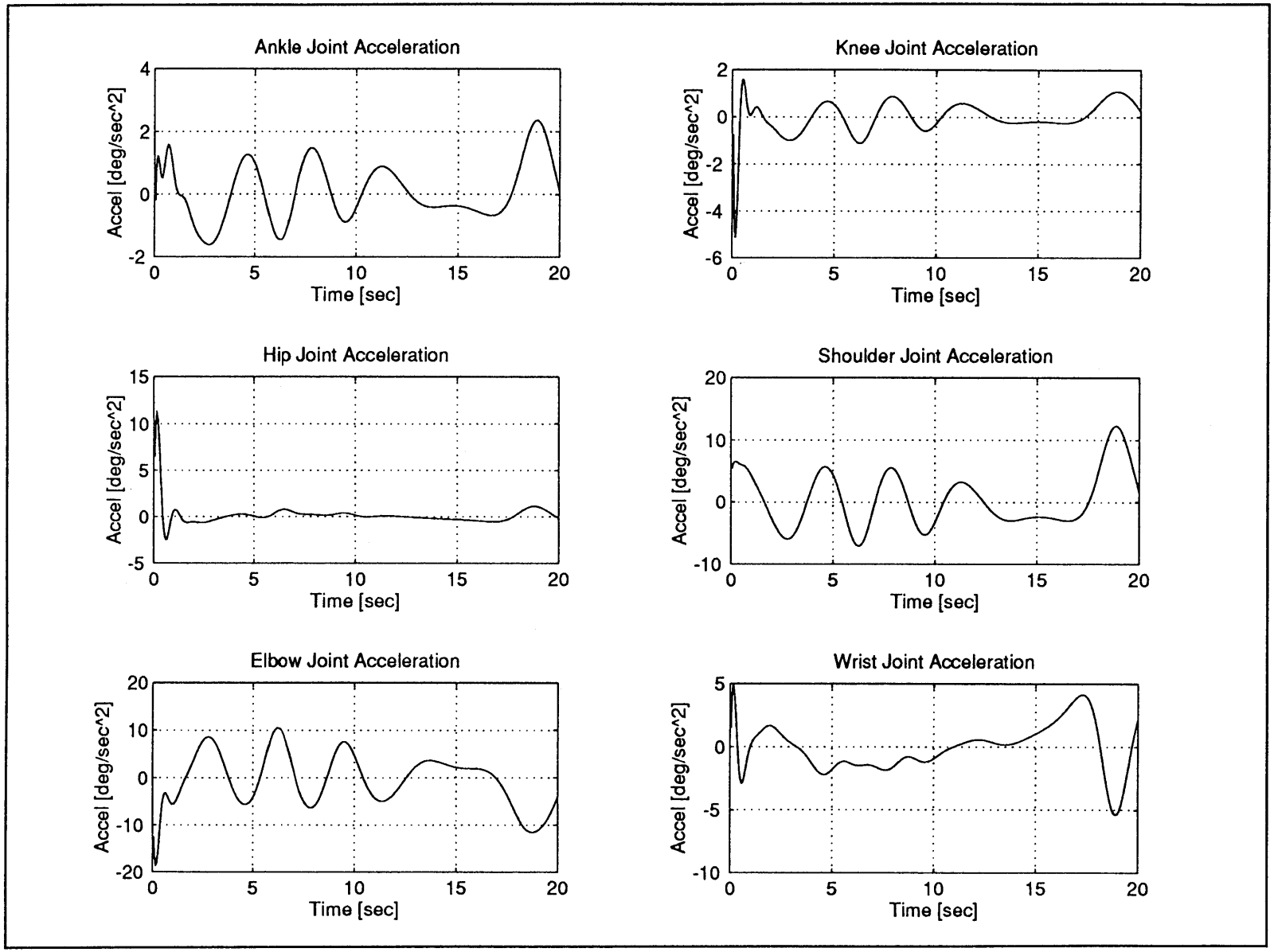


Figure 4.9 Joint acceleration plots for simulation no. 2 (compliant lower body).

Table 4.10 Maximum and minimum accelerations for joints during simulation No. 2 - Compliant Lower Body.

Joint	Time of Max. [sec]	Max. Acceleration [deg/sec^2]	Time of Min. [sec]	Min. Acceleration [deg/sec^2]
Ankle	18.90	2.29	2.70	-1.72
Knee	0.50	1.72	0.20	-4.58
Hip	0.20	10.89	0.60	-2.29
Shoulder	18.80	12.03	6.20	-6.88
Elbow	6.20	10.31	0.20	-18.33
Wrist	0.20	4.58	18.90	-5.16

4.3.2 Inverse Dynamics (No. 2 - Compliant Lower Body)

Joint torque values for the second simulation are shown in Figure 4.10 and the maximum and minimum values are summarized in Table 4.11. A dramatic reduction in torque magnitude is demonstrated for all the joints as compared to the first simulation. Clearly, the speed with which the endpoint trajectory is followed has a large effect on the required joint torques. On average, the magnitudes of the torque values are about one eighth to one tenth of the magnitudes of the torques obtained in the first simulation. The angular velocity at which the circular path is traced out was reduced by one half and the circle radius was reduced by one half so the torque factor of one eighth is approximately predicted by the simple relation

$$f = m\omega^2 r \tag{4.2}$$

where

f = centrifugal force
m = mass of manipulated payload (Spartan)
ω = angular velocity of circular trajectory
r = radius of circle

Since the moment arms over which the centrifugal force (applied at the hand) acts, in relation to the various segments of the system, do not vary to a great extent, the resulting joint torques are approximately proportional to the magnitude of the centrifugal force during both simulations and are thus also approximately scaled by the factor of one eighth.

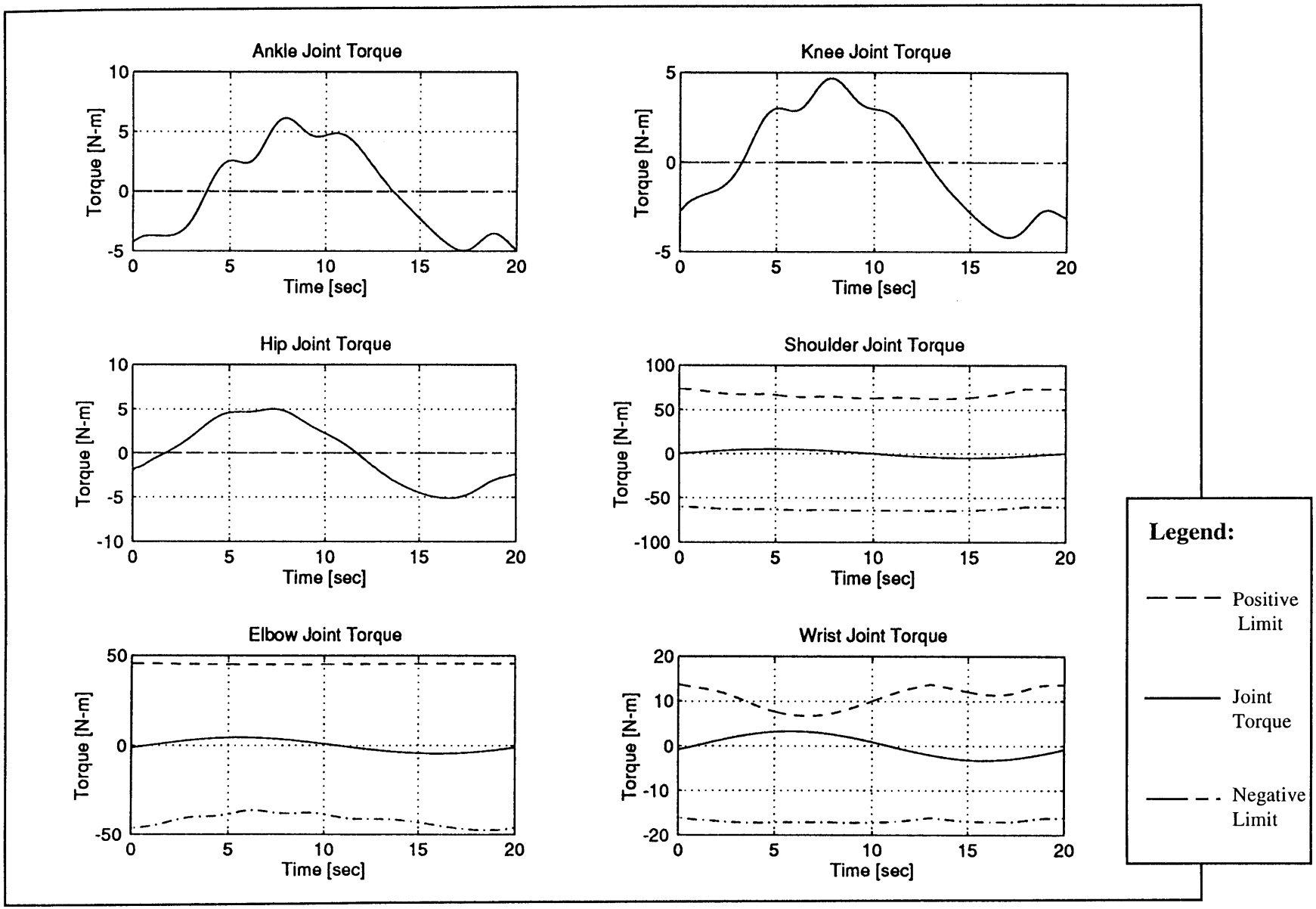


Figure 4.10 Joint torque plots for simulation no. 2 (compliant lower body).

Table 4.11 Maximum and minimum torques for joints during simulation No. 2 - Compliant Lower Body.

Joint	Time of Max. [sec]	Max. Torque [N-m]	Time of Min. [sec]	Min. Torque [N-m]
Ankle	8.00	6.14	17.30	-4.95
Knee	7.80	4.68	17.10	-4.19
Hip	7.30	5.01	16.50	-5.13
Shoulder	4.80	5.04	15.00	-5.14
Elbow	5.50	4.45	16.00	-4.50
Wrist	5.80	3.26	15.80	-3.26

As seen in Figure 4.10 the large reduction in joint torques places this manipulation task within the range of human strength values for the various joints. This fact, combined with the observation that the range of motion of the joints are not significantly exceeded, predicts that the mass manipulation task evaluated in the second simulation can quite comfortably be performed by an astronaut, in contrast to the first simulation's mass handling task, which was well beyond the range of human capability.

Even though the overall joint torques have been greatly reduced, it can be seen that the largest positive torque value occurs in the ankle joint, just as in the case of the first simulation, although slightly larger negative torques occur in the hip and shoulder than in the ankle during this compliant lower body simulation.

4.4 Animation and Data Display

The animation and data display capabilities of the EVADS program are particularly successful. Even though the program is in a rudimentary stage, it has already proven invaluable in interpreting simulation results and in diagnosing problems, especially during the early stages of a simulation.

To visualize the computer representation, consider the two computer screen images shown in Figure 4.11 which were taken at the start of the first simulation. Two similar images for the second simulation are shown in Figure 4.12. One can see at a glance whether the initial configuration corresponds with that desired. If the simulation runs into a snag, such as a singularity, then the configuration causing the singularity is easily identified. In addition, the ability to correlate the animation figure with the data for a given parameter at a specific point in time, along with the availability of the whole time history of that parameter, is very useful in diagnosing the reasons for certain parameters

reaching or exceeding the limits of their nominal range (e.g., a joint angle compared to its physiological range or a joint torque compared to a human strength curve).

The three primary portions of the EVADS display, the animation area, the data plot area, and the parameter selection buttons are shown in Figures 4.11 and 4.12. Two screen images are shown, each one displaying a different viewing angle. Through the use of the pan, rotation, and zoom controls of EVADS, the user can obtain an unlimited number of different viewing angles and scales quite easily. This proves to be particularly useful when certain objects are obscured or when it is desirable to focus in on a specific detail of the animated figure.

The differences between the initial configuration for the two simulations can be noticed by comparing Figures 4.11 and 4.12. In Figure 4.11 the lower body is placed in a neutral weightlessness posture and the hands start out from a point close to shoulder level. In the initial configuration of the second simulation, shown in Figure 4.12, the lower body joints (ankle, knee, and hip) start out in the same neutral body posture, but the arm joint angles are significantly altered. The shoulder joint starts out at zero degrees (i.e., straight down along the side of the trunk), the elbow joint starts at a ninety degree angle, and the wrist joint starts out at zero degrees. All of these angles are measured with respect to the centerline of the preceding body (the shoulder angle is offset by 180° with respect to the trunk so that 0° is with the upper arm pointing down instead of up). Most importantly, it is clear that in the second simulation the wrist joint begins the mass handling task in a more comfortable position close to the center of its range of motion.

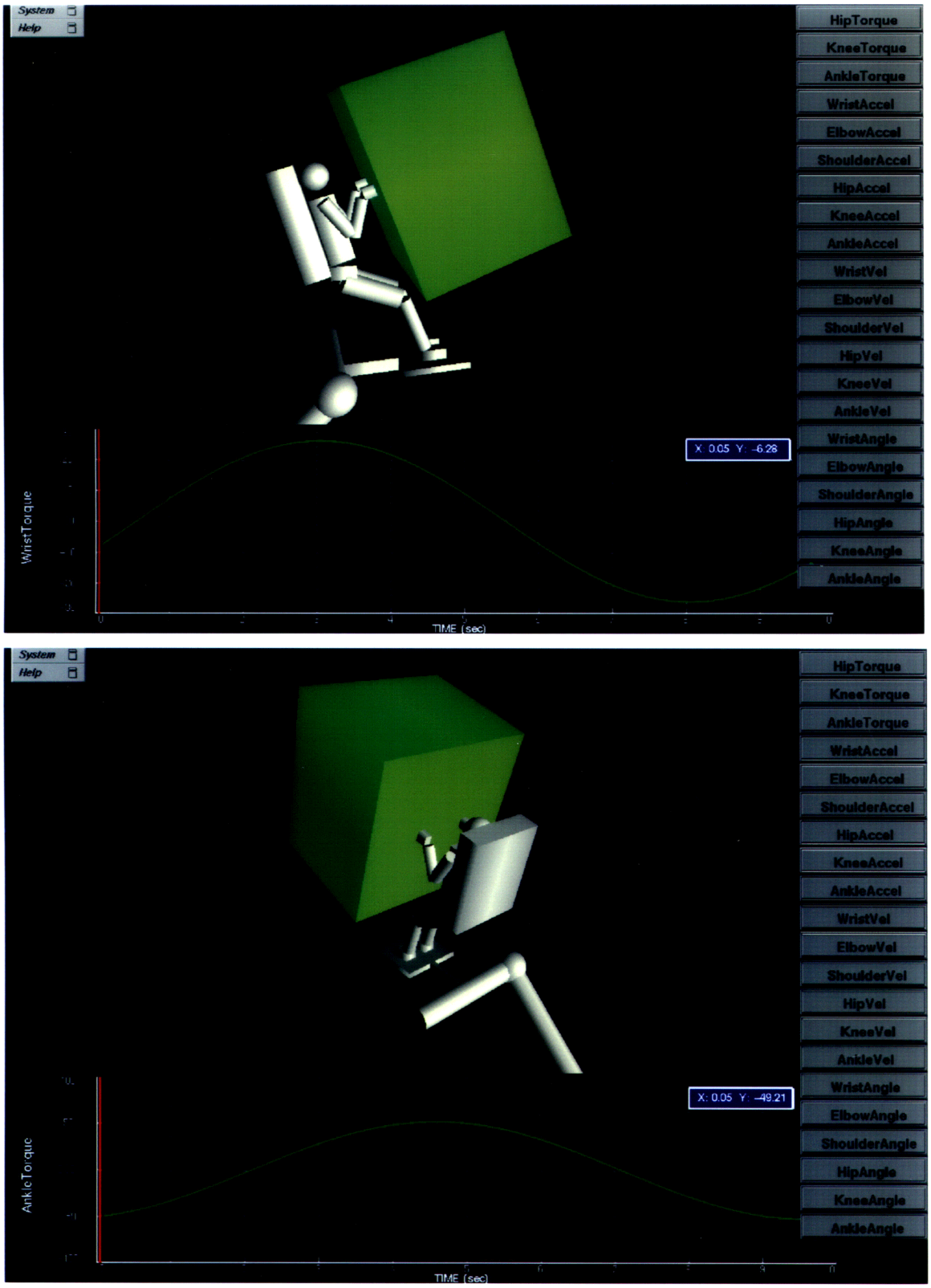


Figure 4.11 Two images of EVADS screen showing different viewing angles of simulation no. 1 – fixed lower body.

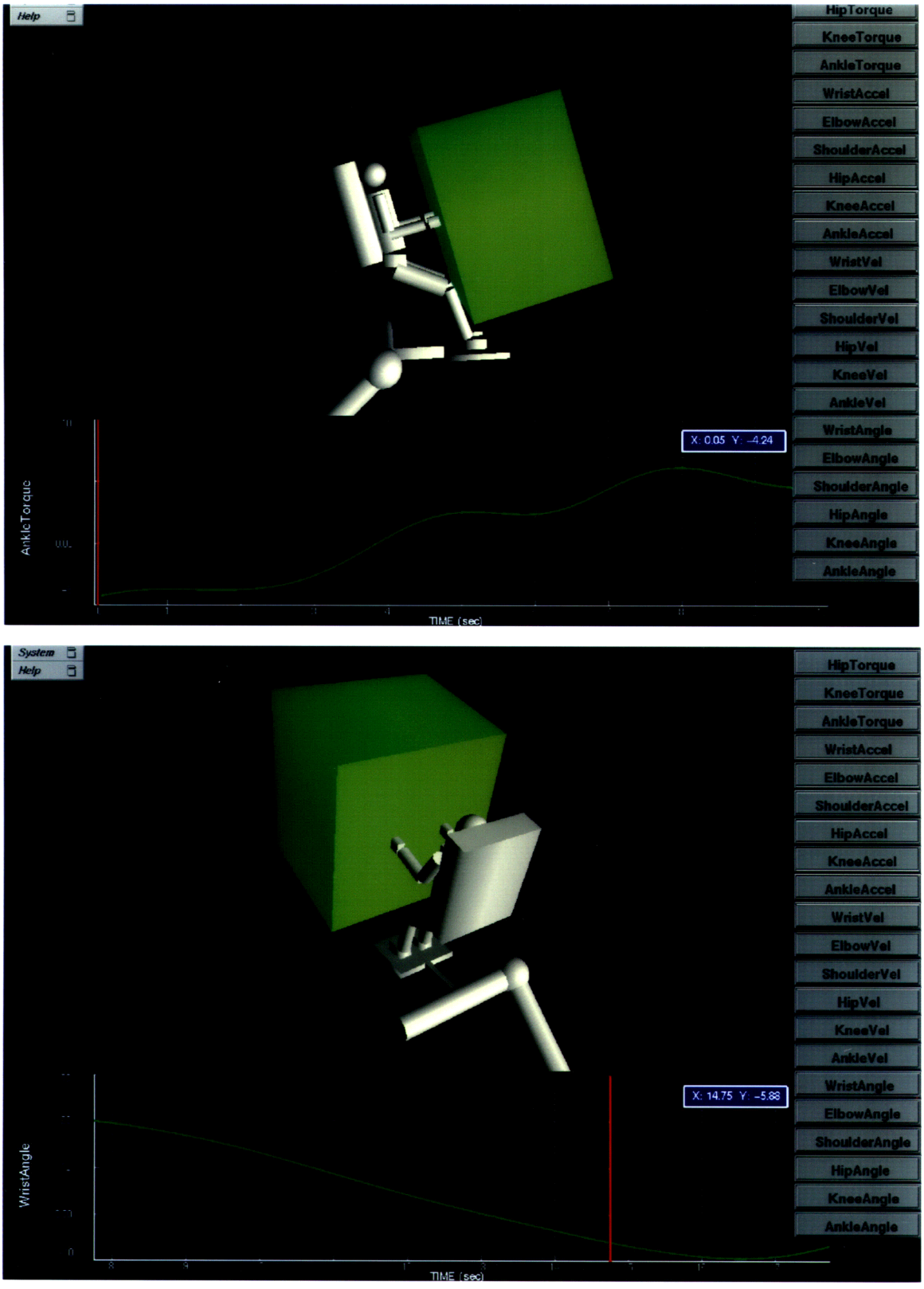


Figure 4.12 Two images of EVADS screen showing different viewing angles of simulation no. 2 – compliant lower body.

5. Discussion and Conclusions

Results of this research effort have been very encouraging. It has been shown that computational simulation of extravehicular activity can overcome many of the limitations of physical simulators and at the same time reveal quantitative information on the dynamic state of the astronaut's body, such as joint torques, that was previously unavailable.

To a certain extent, the research effort described in this thesis was able to take advantage of past work conducted in other areas of application of multibody dynamics such as robotics or sport biomechanics. However, little or no multibody dynamics analysis has been applied to EVA and so it has been a pioneering effort in many ways. For this reason, the emphasis has been on creating a tool for dynamics analysis of EVA, rather than focusing too heavily on the accuracy and detail of the human body models employed. Nevertheless, an effort was made to ensure that the values obtained were realistic and reasonably accurate. For instance, using simplified models of the human body, the values for the hand calculated and computer calculated test torques are shown to be remarkably consistent, deviating by less than 0.03%. Still, certain simplifications should be acknowledged. Rudimentary pin joints replace the complexity of human body joints and mass properties of body segments are calculated from elementary geometric solids (cylinders, blocks, and spheres). In addition, some parameters, such as the lower body spring and damping constants, are estimated based on research and intuition rather than experimental data.

The philosophy adopted in creating simulations has been to start with the simplest possible model that yields non-trivial results. Once this model is working, it is expanded incrementally to incorporate more complexity and additional degrees of freedom and in the process yield more realistic and more accurate results. For instance, to simulate the Spartan mass handling EVA a simple seven segment model of an astronaut manipulating a massive payload has been created. It is assumed that the astronaut's feet are rigidly attached to an object fixed in inertial space. In an actual EVA, the astronaut's feet would most likely be clamped in a Portable Foot Restraint (PFR) attached to the Space Shuttle Orbiter via its robot arm (the Remote Manipulator System, or RMS). In reality, the PFR and RMS are not perfectly rigid and the although the Shuttle Orbiter's mass is very large it will still experience very small accelerations due to the astronaut's motions. Since the dynamic effects of the Orbiter, RMS, and PFR are secondary they have been ignored in

these simulations. The flexibility of the PFR and RMS, and the finite mass of the Orbiter, should be investigated in further research.

Two simulations are carried out, both of which are based on an actual EVA involving mass handling exercises using a Spartan 204 free flying spacecraft. While the second simulation improves on the first, there are certain elements that are common to both. For instance, the manipulation task is specified only in terms of the Cartesian coordinates of the endpoint (hand) of the system, although the kinematics of the system may be subject to certain constraints that direct the way in which the body achieves the motion task, e.g., fixing a joint angle or applying springs and dampers to joints. During the first phase, inverse kinematics, the simulation code "learns" the joint coordinate values of position, velocity and acceleration as the hand c.m. follows the prescribed circular trajectory by storing these values in a two dimensional state-time array. The hand maintains a fixed orientation with respect to "ground", ensuring that the Spartan payload remains at a constant attitude and executes only translational motions. In the second phase, inverse dynamics, the joint kinematics are recalled and used to prescribe the motion of the body and the required joint torques are calculated. The differences between the two simulations are described in the next two sections.

5.1 First Simulation - Fixed Lower Body

In the first simulation it is specified that the ankle, knee, and hip joints must maintain a constant angle for the duration of the run. For this reason, the angular velocity and acceleration of these joints are zero, although non-zero torques are required to hold the joints at the specified angles. The torque in each joint varies with time due to the changing configuration of the arm segments and payload, the motion state, and the torques applied in other joints.

The trajectory followed by the c.m. of the hand is a circle of radius 0.15 m, circumscribed at an angular velocity of 0.628 rad/sec (one revolution in 10 sec). The starting point of the hand c.m. is 0.300 m from the chest and aligned with the shoulder. An undesirable consequence of this starting position and of the size of the circle is that the wrist joint severely exceeds its range of motion when the hand is positioned close to the chest.

In order to maintain the required speed while manipulating a massive object like Spartan (1,201 kg) along the circular trajectory, the body is expected to exert joint torques well beyond the range of human capability. Enhancements to avoid the wrist angle and joint torque problems are addressed in the second simulation.

5.2 Second Simulation - Compliant Lower Body

The main objectives of the second simulation are to keep the wrist joint angular excursions within the limits of the wrist range of motion; to reduce the joint torque levels such that they are below the maximum torque levels achievable by a human; and to improve the overall realism of the simulation.

To avoid the range of motion violations exhibited by the wrist joint in the first simulation, the initial configuration of the system is altered for the arm joints. In this case, the shoulder starts at an angle of 0°, the elbow at 90°, and the wrist at 0° (Recall Figure 4.12). This starting configuration works very well in keeping the wrist angles within the limits, although it does cause a very slight violation of the shoulder joint's lower angular limit near the end of the 20 second simulation.

It was seen that to manipulate the heavy payload along a circular trajectory at the speed required in the first simulation (one revolution in 10 seconds) the astronaut would have to produce joint torques well beyond physiological limits due to the high accelerations of the joints. During the second simulation, a more realistic task is attempted in which the hand's circular trajectory is executed at half the angular velocity used in the first simulation and with half the radius. Fortunately, this task requires torque levels well within the limits of human capability.

The realism is also improved by implementing compliance in the lower body through springs and dampers in the ankle, knee, and hip joints. The animation clearly shows how the astronaut's body initially swings away from the payload in reaction to the forces he exerts as he pushes the payload forward, and then swings toward the payload while pulling the payload backward close to the midway point of the trajectory. It is encouraging that the realism of the simulation was enhanced to such a large extent by the addition of passive springs and dampers in the lower body joints, especially considering that the spring and damping coefficients were estimated. It appears, though, that the spring and damping coefficients were chosen too low based on the appearance of underdamped oscillations in the system together with the fact that a human is capable of producing about four times the torque levels seen in the leg joints. To model the greater stiffness in the lower body joints in the future, higher values for the spring and damping constants should be chosen. However, more data is needed on lower body torque capability before this can be accomplished. Alternatively, a more systematic approach based on control theory applied to human motion, such as optimal control methods, should lead to interesting results.

5.3 EVADS Computer Program

The importance of visual aspects in the ability to improve on successive simulations is clearly demonstrated through the use of the animation and data plotting functions of the EVADS user interface. It is seen that the qualitative information conveyed by the animation can be just as important as the quantitative information conveyed by the plots. Combining these two elements provides the analyst with a very powerful tool for assessing the dynamic effects of EVA operations.

A number of enhancements to the EVADS program are currently being developed. These include: the ability to display more than one curve on a parameter plot and, in particular, to display physiological limits of a parameter along with that parameter (e.g., joint torque limits); the option of displaying multiple plots at the same time so that different parameters can be compared; tracing the trajectory followed by a certain point (e.g., the center of mass of the hand); and improvements to the manner in which button menus are utilized. In addition, a longer range goal is to separate the various portions of EVADS (animation, data plots, button menus) so that each of these can be contained in a window of its own. This would greatly improve the flexibility of the display format.

Ultimately, the desire is to integrate SD/FAST, the dynamic simulation code, and EVADS into one program so that the user can specify the description of the dynamic system, the initial configuration, the trajectory or force application task, the animation, and the data plots, all by means of the convenient user interface offered by EVADS. Viewing the animation and plots of a dynamic system while the simulation code is executing offers the great advantage of the ability to diagnose errors that occur in the midst of the simulation and that sometimes even cause an interruption of the simulation.

5.4 Conclusions

While the simulation code and EVADS graphical interface are clearly still in the developmental stage, it is felt that the main objectives of this stage of the research effort were accomplished. In particular, the primary goal of demonstrating the utility of computational simulation of multibody dynamic systems for EVA research and training was achieved. The specific objectives of this research effort, as listed in chapter 1, were met in the following ways.

1) A convenient means of modeling the dynamic system was demonstrated by means of the seven segment model of the astronaut and the attached payload. This model was described in a system description text file which was used as input to SD/FAST.

2) The system model was transformed into equations of motion by means of the SD/FAST program which represented these equations in an implicit computational form using C code.

3) Computer code written in C language was used to drive the desired simulations by utilizing the functions representing the equations of motion. It was shown that this simulation code is adaptable to a variety of conditions by starting the two simulations from different initial configurations and by making the lower body rigid in the first simulation and compliant in the second simulation.

4) The astronaut motion to be performed was specified in terms of task (endpoint) coordinates alone by means of two virtual sliding joints defined in the X and Y directions of the global coordinate system. The kinematics, in terms of joint coordinates, of the astronaut's body during the manipulation task were successfully calculated using only the endpoint trajectory time history as input.

5) An inverse dynamics computation was successfully performed in each simulation to determine the joint torques using the calculated joint kinematic data (angular position, velocity, and acceleration) as input.

6) A graphics program displaying animation and data plots was successfully created. This program, called EVADS, uses simulation data as input and creates animation that is synchronized with the calculated data plots at the active time point in the plot display area. The user has the ability to select which parameter is displayed, to run the animation in real time, and to step back and forth through the simulation time history.

7) The results of inverse kinematics and inverse dynamics portions of the simulations were interpreted both quantitatively and qualitatively. Consideration of the results produced conclusions about the feasibility and efficiency of the simulated EVA operation and led to practical suggestions about how the actual EVA task might be altered to improve effectiveness. Furthermore, these results gave clues as to how the simulation itself might be improved.

8) Significant improvements in realism and accuracy were accomplished by means of the refinements incorporated in successive simulations. The second simulation successfully solved the problems of wrist joint range of motion violation and unattainable joint torques while the overall realism of body motions was enhanced.

While both of the simulations performed represent highly simplified models of EVA tasks, it is encouraging that significant and revealing results are already obtained by applying the principles of multibody dynamics. Not only does computational simulation provide useful quantitative and qualitative predictions of the dynamic effects involved in a planned EVA, but it can also facilitate the creation of practical concepts for improving the efficiency and comfort of an EVA task. A good example of this arises by observing that the largest joint torques in both simulations occurred in the ankle. Clearly, the mechanical disadvantage that the ankle joint incurs, in terms of both the overall configuration of the body and payload and in terms of its own construction (somewhat of an Achilles' Heel syndrome), is bound to be a source of weakness and inefficiency in any EVA task involving the manipulation of large masses or the exertion of large forces via the hands. The problem could be alleviated by means of a brace mechanism which provides a stiff load path between a cuff around the lower leg (just below the knee) and the Portable Foot Restraint (PFR) or some other foot restraint. The brace would have a telescoping action and pivots at the points of attachment on the lower leg and PFR to allow freedom of movement when unlocked. Turning a simple locking handle, within reach of the astronaut's hand, would rigidize the brace prior to the performance of a manipulation task. The anticipated location and conceptual design of this device is shown in Figure 5.1.

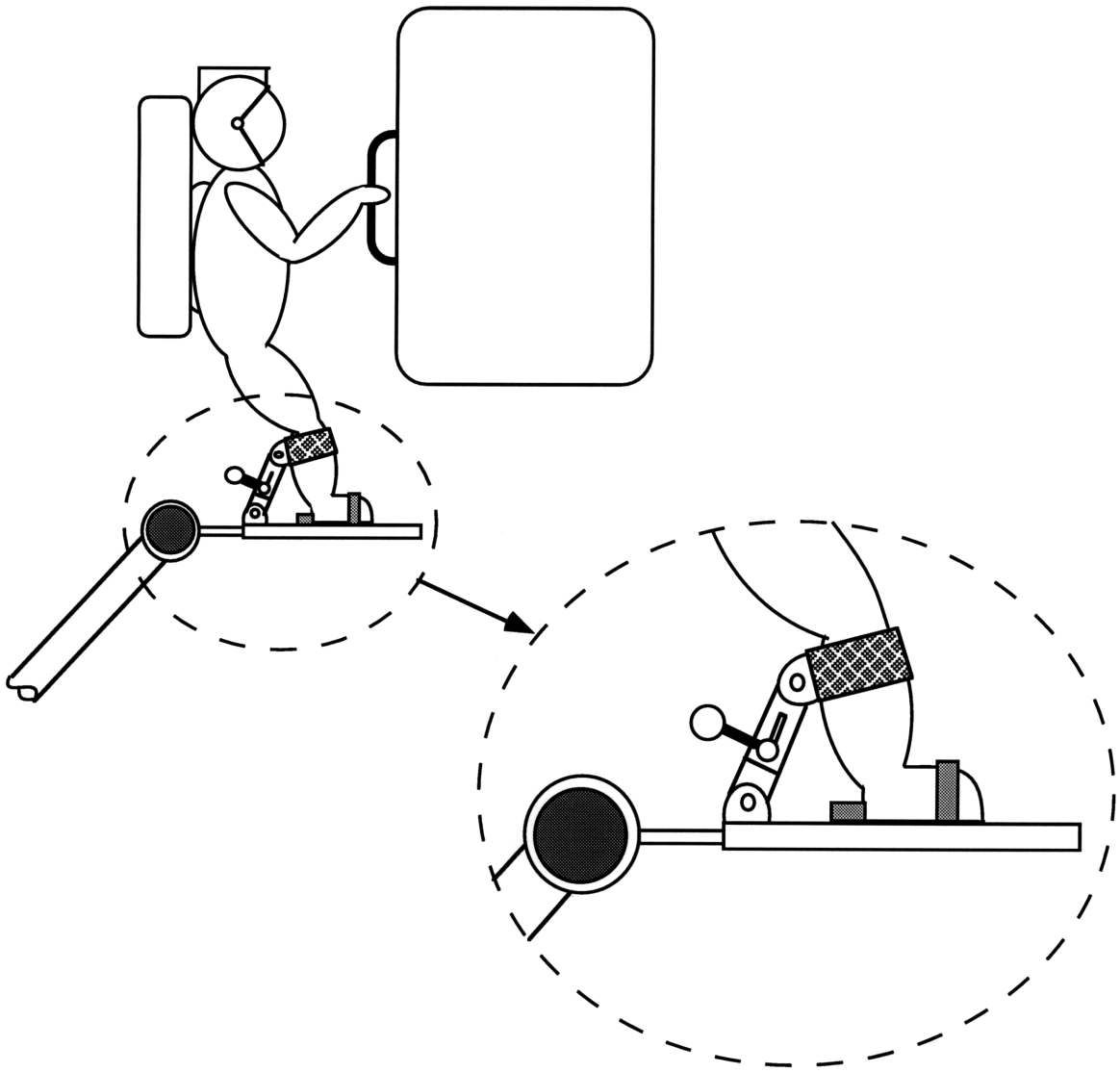


Figure 5.1 Lower leg brace device for reducing ankle torques during EVA mass manipulation tasks.

Although the work of creating the basic EVA dynamic simulation tool has progressed a great deal, there remain many future research activities to pursue; some of which are presented in the next section.

5.5 Recommended Future Research

The most obvious enhancements to the capabilities of the EVA dynamics simulation tool and some research questions to be addressed are summarized below. Greater accuracy of numerical results can be obtained by incorporating more precise data on the mass properties (segment mass, moments of inertia, and products of inertia) of human body segments. This data should be calculated as a function of the overall mass and

dimensions of the human subject being modeled. In addition, more realistic spring and damping constants should be obtained. It would also be helpful to include joint velocity limit curves and joint torque limits not already included.

One of the most important and novel contributions to the multibody system model of the human, in the case of EVA applications, would be the inclusion of the mechanical effects of the spacesuit on astronaut motions. It is anticipated that these effects can be largely modeled by the inclusion of torsional springs in the model's joints and by adding the space suit mass properties to the existing human body segment mass properties. The spacesuit model springs, which would act in parallel with other springs representing the passive and active torques of the human body itself, will likely require nonlinear equation forms to represent them. Furthermore, fabric suits usually exhibit a hysteresis pattern for joint torques that could be modeled. A qualitative sketch of a typical spacesuit joint torque curve is shown in Figure 5.2. There is great difficulty in finding published data on spacesuit joint torques. New experiments to measure spacesuit performance directly and establish data sets or regression equations for predicting joint torque based on position, direction of motion, and perhaps velocity may be necessary.

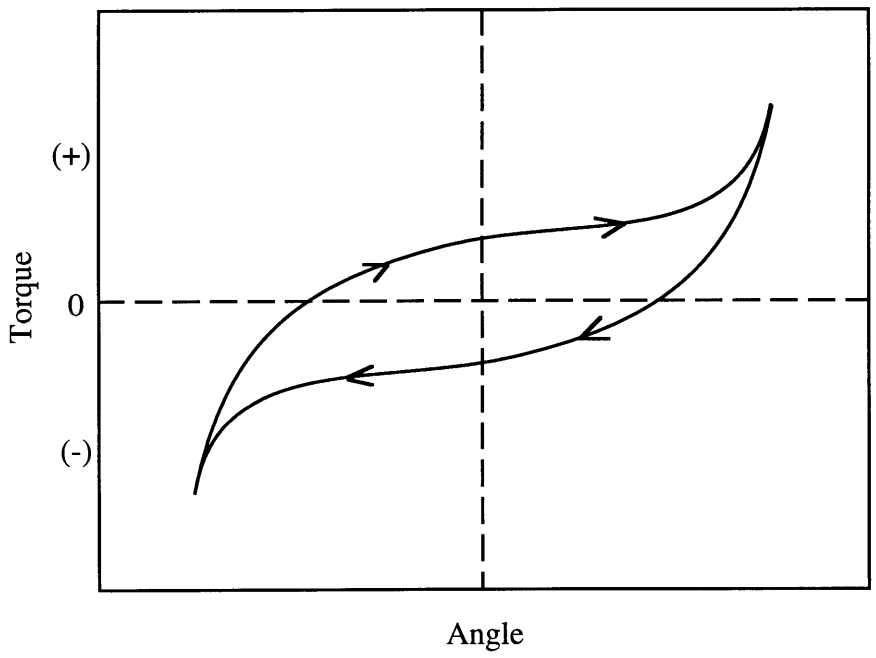


Figure 5.2 Typical spacesuit joint torque curve showing hysteresis effect.

The simulations described in this thesis are restricted to planar motion. It would be very interesting to provide the astronaut model with additional degrees of freedom (e.g., wrist flexion/extension and shoulder adduction/abduction) to enable three-dimensional

motions. The goal of a mass handling task, for instance, would be extended to following a three-dimensional trajectory. One should bear in mind, however, that adding degrees of freedom adds more variables to be evaluated or controlled and the added complexity may make it more difficult to interpret results or control the motion.

Another interesting topic to be investigated is the modeling of the dynamics of the Portable Foot Restraint (PFR), the Remote Manipulator System (RMS), and the Shuttle Orbiter as they relate to EVA tasks. Apparently there is already interest in this issue from NASA EVA training personnel and astronauts. As a first approximation, the Orbiter could be modeled as a single rigid body, the RMS as a three-link multibody system and the PFR as a single link attached to the RMS via a revolute joint with stiff springs.

An issue which needs to be studied further is that of multiple dynamic state solutions to systems with redundant degrees of freedom. In this research effort, solutions have been found by performing a linearized least squares operation. In the future, however, it may be more realistic to find solutions based on some sort of optimization, for instance minimum energy required.

The EVA tasks performed in the two simulations described in this thesis are specified directly by means of computer code. It would be of great use to alter the simulation code so that tasks (e.g., trajectory following, endpoint force exertion, etc.) can be specified through the graphical user interface environment provided by EVADS. This objective falls under the general goal of integrating the simulation code and EVADS program. It would also be worthwhile to develop algorithms that provide spline fits to keypoint trajectory or force data to avoid the user having to specify the EVA task in excessive detail.

An interesting extension of the simulation abilities would be to apply physiological analysis to the data obtained from the dynamical analysis. Kinematic and torque data could theoretically be used to calculate estimates of power, workload, and body temperature (for suit thermal regulation issues). Furthermore, employing physiological formulas might make it possible to estimate metabolic parameters such as O_2 uptake, CO_2 production, heart rate, cardiac output, and so on.

Up to now the control strategies applied in these EVA simulations have been quite rudimentary. An effort should be made to model human control strategies, such as optimal control or the McRuer crossover model (McRuer, Graham et al. 1965), and to include white noise and time delays in the human system.

Finally, but perhaps most importantly, there is a need to perform experimental verification of the simulation results. To mimic weightlessness, experiments could be performed in a neutral buoyancy facility or on a "zero-g" aircraft. Parabolic flight is less

likely to be used for these experiments due to the short duration of continuous weightlessness available (20-25 sec), particularly if the test subject is encumbered by a space suit. Whatever the facility, test subjects will perform mass manipulation of an object along a prescribed trajectory (probably aided by visual cues) while wearing a spacesuit. Tests should be conducted with the subject unsuited as well. The kinematics of the subject's motions (i.e., body segment positions, velocities, and accelerations) will be recorded by some means, for instance a video scan system picking up markers on the subject's spacesuit or body. Alternatively, there may be a possibility of employing a mechanical body motion measurement device, such as those manufactured by the EXOS company in Massachusetts. Unfortunately, there is no practical way of directly measuring the torque in the subject's joints (surgically implanted strain gages attached directly to tendons have been used on animals, but this is not a favored option for human subjects). Instead, the torques can be calculated from the kinematics data using inverse dynamics. Even though the torques themselves cannot be obtained by experiment directly, it is believed that kinematic data will sufficiently characterize the differences and similarities between the theoretical simulation results and the experimental results.

5.6 Summary Paragraph

This preliminary research effort has shown that the application of computational multibody dynamic simulations to extravehicular activity holds great promise as a valuable tool for analysts, trainers, and astronauts. For the first time, it is possible to obtain accurate numerical predictions of quantities such as joint angles and joint torques that are experienced in motion tasks, such as mass manipulation, tool handling, or translation and orientation of an astronaut's body, carried out during extravehicular activity. Having access to this quantitative information, combined with the qualitative information displayed in three-dimensional rendered graphics animation, enables the user to access important factors such as range of motion, exertion, and efficiency and comfort levels. In addition, it has been shown that computational simulation avoids many of the limitations of physical simulators, such as lack of degrees of freedom or friction, and is restricted only by the degree to which the user is willing or able to mathematically describe the physical principles and control laws involved. Furthermore, the convenience, low cost, and quick turnaround time of computational simulations greatly facilitates the process of exploring different scenarios for an EVA task and determining the optimum way in which to perform the desired procedure.

References

Armstrong, W. W. and M. W. Green (1985). "The dynamics of articulated rigid bodies for the purposes of animation." The Visual Computer **1**(4): 231-240.

Asada, H. and J.-J. E. Slotine (1986). Robot Analysis and Control. John Wiley and Sons.

Calderale, P. M. and G. Scelfo (1987). "Mathematical models of musculo-skeletal systems." Engineering in Medicine **16**(3): 131-146.

Cavagna, G. A., A. Zamboni, et al. (1972). "Jumping on the Moon: Power Output at Different Gravity Values." Aerospace Medicine (April): 408-414.

Chace, M. A. (1978). "Using DRAM and ADAMS Programs to Simulate Machinery, Vehicles." Agricultural Engineering (Nov, Dec): 18-19, 16-18.

Davis, B. L. and P. R. Cavanagh (1993). "Simulating reduced gravity: a review of biomechanical issues pertaining to human locomotion." Aviat. Space Environ. Med. **64**(June): 557-66.

Featherstone, R. (1983). "The Calculation of Robot Dynamics Using Articulated-Body Inertias." The International Journal of Robotics Research **2**(1): 13-30.

Fleischer, G. E. and P. W. Likins (1974). Attitude Dynamics Simulation Subroutines for Systems of Hinge-Connected Rigid Bodies. Jet Propulsion Laboratory.

Frisch, H. P. (1974). A Vector-Dyadic Development of the Equations of Motion for N Coupled Rigid Bodies and Point Masses. NASA.

Frisch, H. P. (1975). A Vector-Dyadic Development of the Equations of Motion for N Coupled Flexible Bodies and Point Masses. NASA.

Griffin, M. D. and J. R. French (1991). Space Vehicle Design. Washington, DC, AIAA.

Harwood, W. (1995). Spacewalk Estimate To Build Station Soars. SPACE NEWS. 4, 21.

Haug, E. J. (1989). Computer Aided Kinematics and Dynamics of Mechanical Systems - Volume 1: Basic Methods. Needham Heights, MA USA, Allyn and Bacon.

He, J., R. Kram, et al. (1991). "Mechanics of running under simulated low gravity." J. Appl. Physiol. **71**(3): 000-000.

Hemami, H. (1985). "Modeling, Control, and Simulation of Human Movement." CRC Critical Reviews in Biomedical Engineering **13**(1): 1-34.

Hollars, M. G., D. E. Rosenthal, et al. (1994). SD/FAST User's Manual. Symbolic Dynamics, Inc.

Holloway, T. W. (1992). INTELSAT Capture Lessons Learned. NASA.

Hooker, W. W. (1970). "A Set of r Dynamical Attitude Equations for an Arbitrary n-Body Satellite Having r Rotational Degrees of Freedom." AIAA Journal **8**(July): 1205-1207.

Hooker, W. W. (1974). Equations of Motion for Interconnected Rigid and Elastic Bodies: A Derivation Independent of Angular Momentum. Lockheed Missiles and Space Co.

Hooker, W. W. and G. Margulies (1965). "The Dynamical Attitude Equations for an n-Body Satellite." The Journal of the Astronautical Sciences **XII**(4): 123-128.

Isaacs, P. M. and M. F. Cohen (1987). "Controlling Dynamic Simulations with Kinematic Constraints, Behavior Functions, and Inverse Dynamics." Computer Graphics **21**(4, July): 215-224.

Isaacs, P. M. and M. F. Cohen (1988). "Mixed methods for complex kinematic constraints in dynamic figure animation." The Visual Computer **4**: 296-305.

Kane, T. R. and D. A. Levinson (1983). "Multibody Dynamics." Journal of Applied Mechanics **50**(December): 1071-1078.

Kane, T. R. and D. A. Levinson (1983). "The Use of Kane's Dynamical Equations in Robotics." The International Journal of Robotics Research **2**(3): 3-21.

Kane, T. R. and D. A. Levinson (1985). DYNAMICS: Theory and Applications. McGraw-Hill, Inc.

Kane, T. R., P. W. Likins, et al. (1983). Spacecraft Dynamics. New York, McGraw-Hill.

Kinzel, G. L. and L. J. Gutkowski (1983). "Joint Models, Degrees of Freedom, and Anatomical Motion Measurements." Journal of Biomechanical Engineering **105**(February): 55-62.

Kozloski, L. D. (1994). U.S. Space Gear : outfitting the astronaut. Smithsonian Institution.

Marshall, R. N., R. K. Jensen, et al. (1984). "A General Newtonian Simulation of an N-Segment Open Chain Model." Journal of Biomechanics **18**(5): 359-367.

McDonnell-Douglas (1994). The McDonnell Douglas Human Modeling System MDHMS Version 2.2.

McMahon, T. A. (1984). Muscles, Reflexes, and Locomotion. Princeton, NJ, Princeton University Press.

McRuer, D. T., G. Graham, et al. (1965). Human pilot dynamics in compensatory systems - theory, models and experiments with controlled element and forcing function variations. Air Force Flight Dynamics Laboratory, Wright-Patterson AFB, Ohio.

NASA (1987). Man-Systems Integration Standards.

NASA (1992). Space Shuttle Mission STS-49 Press Kit, Maiden Voyage of Endeavour.

Newman, D. J. and H. L. Alexander (1991). Human locomotion and workload for simulated lunar and martian environments. 42nd Congress of the International Astronautical Federation, Montreal, Canada, International Astronautical Federation.

Newman, D. J. and M. Barratt (1995). Chapter 22: Life Support and Performance Issues for Extravehicular Activity (EVA). Introduction to Space Life Sciences. Kleger Press, Orion Books (now in press).

Pandy, M. G., F. E. Zajac, et al. (1990). "An Optimal Control Model for Maximum-Height Human Jumping." J. Biomechanics **23**(12): 1185-1198.

Pandya, A. K., S. M. Hasson, et al. (1992). Correlation and Prediction of Dynamic Human Isolated Joint Strength From Lean Body Mass. NASA.

Pandya, A. K., J. C. Maida, et al. (1992). The Validation of a Human Force Model To Predict Dynamic Forces Resulting From Multi-Joint Motions. NASA.

Price, L. R., M. A. Fruhwirth, et al. (1994). Computer Aided Design and Graphics Techniques for EVA Analysis. 24th International Conference on Environmental Systems and 5th European Symposium on Space Environmental Control Systems, Friedrichshafen, Germany, Society of Automotive Engineers, Inc.

Ramey, M. R. and A. T. Yang (1980). "A Simulation Procedure for Human Motion Studies." Journal of Biomechanics **14**(4): 203213.

Reinhardt, A. (1989). Results and Applications of a Space Suit Range-of-Motion Study. 19th Intersociety Conference on Enviromental Systems, San Diego, CA, Society of Automotive Engineers, Inc.

Roberson, R. E. and J. Wittenburg (1966). A Dynamical Formalism for an Arbitrary Number of Interconnected Rigid Bodies, With Reference to the Problem of Satellite Attitude Control. The Third Congress of the International Federation of Automatic Control, London,

Rosenthal, D. E. and M. A. Sherman (1986). "High Performance Multibody Simulations via Symbolic Equation Manipulation and Kane's Method." The Journal of the Astronautical Sciences **34**(3, July-September): 223-239.

Schaechter, D. B. and D. A. Levinson (1988). Interactive Computerized Symbolic Dynamics for the Dynamicist. 1988 American Control Conference, Atlanta, GA USA, American Automatic Control Council (IEEE).

Scher, M. P. and J. R. Kane (1969). Alteration of the state of motion of a human being in free fall. Stanford University, Division of Applied Mechanics.

Silver, W. M. (1982). "On the Equivalence of Lagrangian and Newton-Euler Dynamics for Manipulators." The International Journal of Robotics Research **1**(2): 60-70.

Singh, R. P. and R. J. Vandervoort (1985). "Dynamics of Flexible Bodies in Tree Topology/a Computer-Oriented Approach." Journal of Guidance, Control and Dynamics **8**(5):

Steiner, M. D. and C. H. Seaman (1994). Extravehicular Activity Annex - Spartan 204. NASA.

Walker, M. W. and D. E. Orin (1982). "Efficient Dynamic Computer Simulation of Robotic Mechanisms." Journal of Dynamic Systems, Measurement, and Control **104**(September): 205-211.

Webbon, B. (1994). Human Requirements for Life Support and Extravehicular Activity in Space. Ames Research Center, NASA.

Wilhelms, J., M. Moore, et al. (1988). "Dynamic animation: interaction and control." The Visual Computer **4**: 283-295.

Wittenburg, J. (1979). The Dynamics of the Human Body. The Fifth World Congress on Theory of Machines and Mechanisms, ASME.

Wortz, E. C. and E. J. Prescott (1966). "Effects of Subgravity Traction Simulation on the Energy Costs of Walking." Aerospace Medicine (December): 1217-1222.

Appendix A - System Description File

Below is the text file used as input to SD/FAST which describes the geometry, degrees of freedom, mass properties, and interconnection of segments in the multibody system model of the EVA crewmember plus the Spartan payload.

```
# File: eva4.sd

# Author: Grant Schaffner        Last Revision: April 25, 1995

# This is an SD/FAST input file describing the multi-body dynamic model of an
# astronaut for the purposes of simulating EVA dynamics.

# The astronaut's body is modeled by an 7-link, 7-dof dynamic system.
# All joints are modeled by 1-dof pins with their axes aligned with the
# +Z axis. The origin of the ground ref frame is at the ankle joint. An 8th
# segment represents a large mass (Spartan payload) to be manipulated that
# is welded to the hand.

# This model does not account for the inertial, elastic, or damping influences
# of a spacesuit.

# No gravity term is specified since this is assumed to take place in
# weightlessness.

# The parameters given below are rough approximations of human body mass
# properties from Frohlich, C., 1979
# Mass properties for Spartan are obtained from NASA JSC documents.
# Units are SI, density = 1000 kg/m^3 (water)

# Note: Both arms and both legs have been combined into a single limb, each
# with double the original density and thus double the mass and double the
# moments of inertia of each individual limb.

# lower leg = cylinder, r=.055, h=.430 (2x)
body = lleg inb = $ground joint = pin prescribed = ?
  mass = 8.172                    inertia = .132096 .012360 .132096
  bodytojoint = 0 -.215 0   inbtojoint = 0 0 0 pin =  0 0 1

# upper leg = cylinder, r=.080, h=.430 (2x)
body = uleg inb = lleg joint = pin prescribed = ?
  mass = 17.300                   inertia = .294244 .055360 .294244
  bodytojoint = 0 -.215 0   inbtojoint = 0 .215 0       pin = 0 0 1

# trunk = block, w=.180, h=.600, d=.300
body = trunk inb = uleg joint = pin prescribed = ?
  mass = 32.400                   inertia = 1.215000 .330480 1.059480
  bodytojoint = 0 -.300 0   inbtojoint = 0 .215 0       pin = 0 0 1

# head = sphere, r=.11
body = head inb = trunk joint = weld
  mass = 5.575                    inertia = .026983 .026983 .026983
```

```
  bodytojoint = 0 -.110 0    inbtojoint = 0 .300 0

#****************************************

# upper arm = cylinder, r=.050, h=.300 (2x)
body = uarm inb = trunk joint = pin prescribed = ?
  mass = 4.712                    inertia = .038286 .005890 .038286
  bodytojoint = 0 0.150 0   inbtojoint = 0 .275 0        pin = 0 0 1

# fore arm = cylinder, r=.045, h=.28 (2x)
body = farm inb = uarm joint = pin prescribed = ?
  mass = 3.562                    inertia = .025074 .003606 .025074
  bodytojoint = 0 0.140 0   inbtojoint = 0 -0.150 0              pin = 0 0 1

# hand = block, h=0.1103, w=.0858, d=.0553 (2x)

# It is desired to have the com of the hand follow a prescribed trajectory.
# This is achieved as follows: Two massless bodies are defined handx & handy.
# handx is connected to ground (at the foot) via a slider in the x-dirn.
# handy is connected to handx by means of a slider in the y-dirn. The real
# hand segment is divided into two halves which have the same geometry and half
# the density of the complete hand. harm is connected to farm by a pin, and
# hlside is connected to handy by a pin. Prescribed motion is set for hslide
# if it is desired to control the orientation of the hand. Finally the two
# halves of the hand are welded together to complete a loop joint.

body = handx inb = $ground joint = slider prescribed = ?
  mass = 0                        inertia = 0 0 0
  bodytojoint = 0 0 0             inbtojoint = 0 .79985 0    pin = 1 0 0

body = handy inb = handx joint = slider prescribed = ?
  mass = 0                        inertia = 0 0 0
  bodytojoint = 0 0 0             inbtojoint = 0 0 0         pin = 0 1 0

body = harm inb = farm joint = pin prescribed = ?
  mass = .523                     inertia = .000664 .000451 .000851
  bodytojoint = 0 0.05515 0       inbtojoint = 0 -0.140 0            pin = 0 0 1

body = hslide inb = handy joint = pin prescribed = ?
  mass = .523                     inertia = .000664 .000451 .000851
  bodytojoint = 0 0 0             inbtojoint = 0 0 0         pin = 0 0 1

# Spartan deployable payload. +Z axis of spartan is aligned with -X axis of
# gnd. +X axis of Spartan aligned with -Y axis of gnd. Products of inertia
# considered insignificant for preliminary simulations.
body = spartan inb = harm joint = weld
  mass = 1201.3907
  inertia =          334.8785          325.8123          352.3724
  bodytojoint = 0 .6220 0   inbtojoint = 0 -.05515 0

# Loop joint : weld two halves of hand together.
body = hslide inb = harm joint = weld
  bodytojoint = 0 0 0       inbtojoint = 0 0 0
  pin = 0 0 1               bodypin = 0 0 1
  inbref = 0 1 0    bodyref = 0 1 0
```

Appendix B - Simulation Code

Simulation code written in C language is provided in this appendix. Only the code for the second simulation is presented here because this file includes the same routines used in the first simulation as well as other routines such as those for the springs and dampers used in the lower body joints.

```
/*****************************************************************************

 eva4

 Grant Schaffner, May 8, 1995

 This simulation is modeled on the Spartan 204 mass manipulation EVA performed
 on STS-63.

 An inverse dynamics analysis performed on a seven link model of an
 astronaut manipulating a massive object (eighth segment) around a circular
 trajectory. The ankle, knee, and hip joints are actuated by passive springs
 and dampers to provide compliance in the lower body.

 There are three phases: (1) Initial Assembly and Velocity analysis;
 (2) Inverse Kinematics; and (3) Inverse Dynamics.

*****************************************************************************/

#include <stdio.h>
#include <math.h>
#include "strength_eva.h"

/* Bodies (Frames) */
#define GROUND -1
#define LLEG   0
#define ULEG   1
#define TRUNK  2
#define HEAD   3
#define UARM   4
#define FARM   5
#define HANDX  6
#define HANDY  7
#define HARM   8
#define HSLIDE 9
#define SPARTAN 10

/* State Variables */
#define NQ  9
#define NU  9
#define NEQ (NQ+NU)
#define NJNT 12

/* Constraints */
#define NC  15
```

```
/* Integration Parameters */
#define DT .05
#define TOL 1e-7
#define CTOL 1e-5
/* #define NSTEP 200 */ /* 10sec (one rev) divided by .05 (dt) = 200 */
#define NSTEP 400  /* 20sec (one rev) divided by .05 (dt) = 400 */

/* Joint Range of Motion (in degrees) */
#define ANKLEMIN -40.0
#define ANKLEMAX  40.0
#define KNEEMIN   0.0
#define KNEEMAX  120.0
#define HIPMIN   -70.0
#define HIPMAX    0.0
#define SHLDRMIN  0.0
#define SHLDRMAX 180.0
#define ELBOWMIN  0.0
#define ELBOWMAX 130.0
#define WRISTMIN -28.3 /* suit = 85% of nude average */
#define WRISTMAX  22.8 /* ditto */


/* Define external symbolic constants  */


/* Declare external variable types */
double omega, rcirc, xstart, ystart, pi, dtr, theta;
double ystore[NSTEP][NEQ], accel[NSTEP][NQ];
double time,state[NEQ];
char pfile[20]="eva4.pos", vfile[20]="eva4.vel", afile[20]="eva4.acc";
char jtfile[20]="eva4.trq", pmxfile[20]="eva4.pmx", pmnfile[20]="eva4.pmn";
char tmxfile[20]="eva4.tmx", tmnfile[20]="eva4.tmn";
FILE *pfile_ptr, *vfile_ptr, *afile_ptr, *jtfile_ptr;
FILE *pmnfile_ptr, *pmxfile_ptr, *tmnfile_ptr, *tmxfile_ptr;
char *body[9] = {"lleg","uleg","trunk","uarm","farm","handx","handy","harm",
    "hslide"};

/* Function prototypes */

void printvals(int nval, double array[]);

/*-------------------------------------------------------------------------*/
/*                    Main program                                         */
/*-------------------------------------------------------------------------*/

main ()
{

/* Declare variables local to main. */

int i, lock[NU], fcnt, err;
int j, flag;
double t, y[NEQ], yinit[NEQ], dy[NEQ], qd[NQ], ud[NU];
double torq_lleg, torq_uleg, torq_trunk, torq_uarm, torq_farm, torq_harm;
double tmn[NQ], tmx[NQ];

double com[3], pos[3], vel[3];
```

```
double perrs[NC], verrs[NC];

double q1,q2,q3,q4,q5,q6;

/* Reminder about external variables. */

extern double omega, rcirc, xstart, ystart;
extern double pi, dtr, theta;

/*************************************************************************
  The section of code below is adapted from the main part of lean.c
  written by Abilash Pandya and Jim Maida (NASA JSC).
*************************************************************************/


/*********************************************************
type definitions for CoeffTable data structure see strength_eva.h.

Decription of force table data structure.

table[joint].function[axis][dir]
   force functions associated with a joint, these
   are indexed by the axis number and the direction
   code.

table[joint].joint_name
   name of joint 80 characters max.

table[joint].axis[axis]
   table of axis numbers (DOF's) for the joint.

table[joint].axis_count
   number of axes or DOF's for the joint.

function[axis][dir].coeff[vid][coeff]
   force function coeffcients index by the velocity code (vid)
   (see velocity code) and coefficient index (1-4).

function[axis][dir].velocity[vid]
   velocity value associated with a given velocity code (vid).

function[axis][dir].eq_count
   number of force function equations associated with
   a given axis and direction.

function[axis][dir].direction_name
   name of a given direction, i.e. extension, flexion, etc.

*********************************************************/

CoeffTable pop_table[MAXJOINT];   /*normalized population coefficients*/

CoeffTable lean_table[MAXJOINT];  /*relationship: mean torque to lean mass*/

CoeffTable ind_table[MAXJOINT];   /*Calculated coeff table based on
                                    mass and %body fat*/

float bf;                  /* body fat percentage */
```

```
float mass;                    /* mass (kg) */


printf("\n _____");
printf("\n|                    |");
printf("\n|   Simulation EVA4.    |");
printf("\n|_____|\n\n");


/*read the two tables 1. Population table  2. lean body table*/
ReadCoeffTable ( pop_table , "ntc" , JC );

ReadCoeffTable ( lean_table, "lbc", JC  );
/*
    printf ("Input %bf (0 - 1) and mass:");
    scanf ("%f %f", &bf, &mass);
*/
    bf=0.06;
    mass=77.11;

    InitJointCoeffTable ( ind_table );
    CalculateIndividualCoeffTable (ind_table, pop_table, lean_table,
                                   bf, mass, JC );
    printf("\n");

/********** End of lean.c adaptation. ********************************/

com [0] = 0.0; pos[0] = 0.0; vel[0] = 0.0;
com [1] = 0.0; pos[1] = 0.0; vel[1] = 0.0;
com [2] = 0.0; pos[2] = 0.0; vel[2] = 0.0;

pi = acos(-1.0);
dtr = pi/180.0;

/* Radius of the  circle and the angular velocity for the circular
   trajectory. */

/*  omega = 2.0*pi/10.0;
  rcirc = 0.15; */
  omega = 2.0*pi/20.0;
  rcirc = 0.075;

/*    printf("Enter filename for position data: ");
    scanf("%s",pfile);

    printf("Enter filename for velocity data: ");
    scanf("%s",vfile);

    printf("Enter filename for acceler. data: ");
    scanf("%s",afile);
*/
    pfile_ptr = fopen(pfile,"w");
    if(pfile_ptr == NULL) {
          printf("Oops. Unable to open file. Exiting...\n");
          exit(-1);
          }
```

```
    pmnfile_ptr = fopen(pmnfile,"w");
    if(pmnfile_ptr == NULL) {
            printf("Oops. Unable to open file. Exiting...\n");
            exit(-1);
            }

    pmxfile_ptr = fopen(pmxfile,"w");
    if(pmxfile_ptr == NULL) {
            printf("Oops. Unable to open file. Exiting...\n");
            exit(-1);
            }

    vfile_ptr = fopen(vfile,"w");
    if(vfile_ptr == NULL) {
            printf("Oops. Unable to open file. Exiting...\n");
            exit(-1);
            }

    afile_ptr = fopen(afile,"w");
    if(afile_ptr == NULL) {
            printf("Oops. Unable to open file. Exiting...\n");
            exit(-1);
            }
/*
    printf("Enter filename for Joint Torques: ");
    scanf("%s",jtfile);
*/
    jtfile_ptr = fopen(jtfile,"w");
    if(jtfile_ptr == NULL) {
            printf("Oops. Unable to open file. Exiting...\n");
            exit(-1);
            }

    tmnfile_ptr = fopen(tmnfile,"w");
    if(tmnfile_ptr == NULL) {
            printf("Oops. Unable to open file. Exiting...\n");
            exit(-1);
            }

    tmxfile_ptr = fopen(tmxfile,"w");
    if(tmxfile_ptr == NULL) {
            printf("Oops. Unable to open file. Exiting...\n");
            exit(-1);
            }

  sdinit();
/*-------------------------------------------------------------------------*/
/* Test three arm joints individually - for Inverse Dynamics verification. */
/*
  wristtest();
  elbowtest();
  shouldertest();
*/
/* exit(); */
/*-------------------------------------------------------------------------*/

 /* Setup initial conditions - body positions and velocities */
```

```
  printf("\nInitial Conditions.....");

/* Initial Configuration */

  for (i = 0 ; i < NEQ ; i++) yinit[i] = 0.0;

/* Assign initial angles to variables to simplify computation of xstart
   and ystart. Angles are measured w.r.t centerline of preceding segment,
   positive in a ccw direction. */

  q1 = 111.0*dtr; /* ankle    */ /* angle wrt RHS horizontal */
  q2 =  47.0*dtr; /* knee     */
  q3 = -52.0*dtr; /* hip      */
  q4 = 180.0*dtr; /* shoulder */
  q5 =  90.0*dtr; /* elbow    */
  q6 =   0.0*dtr; /* wrist    */

  yinit[sdindx(LLEG,0)] = q1-90.0*dtr; /* ref. pos. is upward vertical */
  yinit[sdindx(ULEG,0)] = q2;
  yinit[sdindx(TRUNK,0)] = q3;

  yinit[sdindx(UARM,0)] = q4-180.0*dtr; /* ref. pos. is downward vertical */
  yinit[sdindx(FARM,0)] = q5;
  yinit[sdindx(HARM,0)] = q6;

/*  yinit[sdindx(HSLIDE,0)] = -74.0*dtr; */ /* ?? */

/*  xstart = -0.43*cos(69.0*dtr)-0.43*cos(22.0*dtr)-0.575*cos(74.0*dtr)
          +0.300*cos(51.0*dtr)+0.28*cos(79.0*dtr)+0.05515*cos(20.0*dtr);
   ystart = 0.43*sin(69.0*dtr)+0.43*sin(22.0*dtr)+0.575*sin(74.0*dtr)
          -0.300*sin(51.0*dtr)+0.28*sin(79.0*dtr)+0.05515*sin(20.0*dtr)
          -0.79985; */

  xstart = 0.43*cos(q1)+0.43*cos(q1+q2)+0.575*cos(q1+q2+q3)
          +0.300*cos(q1+q2+q3+q4)+0.28*cos(q1+q2+q3+q4+q5)
          +0.05515*cos(q1+q2+q3+q4+q5+q6);
  ystart = 0.43*sin(q1)+0.43*sin(q1+q2)+0.575*sin(q1+q2+q3)
          +0.300*sin(q1+q2+q3+q4)+0.28*sin(q1+q2+q3+q4+q5)
          +0.05515*sin(q1+q2+q3+q4+q5+q6)-0.79985;
/*
  printf("\nxstart:\t\t%f", xstart);
  printf("\nystart:\t\t%f\n", ystart);
*/
  yinit[sdindx(HANDX,0)] = xstart;
  yinit[sdindx(HANDY,0)] = ystart;

/* Start with prescribed motion ON at handx, handy, and hslide (fixes hand),
   and OFF at lleg, uleg, trunk, uarm, farm, and harm. */

  sdpres(HANDX,0,1);
  sdpres(HANDY,0,1);
  sdpres(HSLIDE,0,1);

  sdpres(LLEG,0,0);
  sdpres(ULEG,0,0);
  sdpres(TRUNK,0,0);

  sdpres(UARM,0,0);
```

```
  sdpres(FARM,0,0);
  sdpres(HARM,0,0);

  for (i = 0 ; i < NEQ ; i++) y[i] = yinit[i];

  printf("done.\n"); /* end of initial conditions */

/*  printf("\nInitial State Vector:\n\tq\t\tu\n");
  for (i=0 ; i<NQ ; i++) printf("%s\t%f\t%f\n",body[i],y[i],y[NQ+i]);
*/
  t = 0.0;

/* Do position and velocity analysis */

  printf("Assembly and Velocity Analysis.....");

  for (i = 0 ; i < NU ; i++) lock[i] = 0;

/* Lock the position and velocity for handx, handy hslide, lleg, uleg, trunk
   and head to the passed-in values. */

  lock[HANDX] = 1;
  lock[HANDY] = 1;
  lock[HSLIDE] = 1;

  lock[LLEG] = 1;
  lock[ULEG] = 1;
  lock[TRUNK] = 1;

/* Optionally, lock the angular position of the arm instead. */

/*  lock[UARM] = 1;
  lock[FARM] = 1;
  lock[HARM]  = 1;  */

/* This function returns compatible position values in the state vector. */
/* Note: sdassemble calls sdumotion to obtain the prescribed motions. */

  sdassemble(t, y, lock, CTOL, 5000, &fcnt, &err);
  if (err != 0){
     printf("\a\n Assembly failed! Error no.: %d\n",err);
     exit();
    }

/* This function returns compatible velocity values in the state vector. */
/* Note: sdinitvel calls sdumotion to obtain the prescribed motions. */

  for (i = 0 ; i < NU ; i++) lock[i] = 0;
  sdinitvel(t, y, lock, CTOL, 5000, &fcnt, &err);
  if (err != 0) printf("\a\n Velocity analysis failed! Error no.: %d\n",err);

/* Check if the position and velocity errors are below CTOL. */

  sdperr(perrs);

/*  printf("\nPosition errors for the NC constraints (ctol=%f):\n",CTOL);
  printvals(NC,perrs); */
```

```
  sdverr(verrs);

/*  printf("\nVelocity errors for the NC constraints (ctol=%f):\n",CTOL);
  printvals(NC,verrs); */

  for (i = 0 ; i < NU ; i++) lock[i] = 0;

/* Print the whole state vector as a check. (optional) */
/*  printf("\nState Vector after assembly and velocity analysis:\n\tq\t\tu\n");
  for(i=0 ; i < NQ ; i++) printf("%s\t%f\t%f\n",body[i],y[i],y[NQ+i]); */

/* Check the arm position. (optional) */
/*
  printf("\nBody generalized coordinates (relative q\'s)\n");
  printf("\nAnkle Angle (deg):\t\t%f", y[sdindx(LLEG,0)]/dtr);
  printf("\nKnee Angle (deg):\t\t%f", y[sdindx(ULEG,0)]/dtr);
  printf("\nHip Angle (deg):\t\t%f", y[sdindx(TRUNK,0)]/dtr);
  printf("\nShoulder Angle (deg):\t\t%f", y[sdindx(UARM,0)]/dtr);
  printf("\nElbow Angle (deg):\t\t%f", y[sdindx(FARM,0)]/dtr);
  printf("\nHand Angle wrt FArm (deg):\t%f", y[sdindx(HARM,0)]/dtr);
  printf("\nHand Angle wrt y (deg):\t\t%f", y[sdindx(HSLIDE,0)]/dtr);
  printf("\n\n");

  sdpos(UARM,com,pos);
  printf("Upper Arm COM location:\t");
   printvals(3,pos);
  sdpos(FARM,com,pos);
  printf("Fore Arm COM location:\t");
  printvals(3,pos);
  sdpos(HARM,com,pos);

  printf("Hand COM location:\t");
  printvals(3,pos);
  printf("\n");

  sdvel(UARM,com,vel);
  printf("Upper Arm COM Velocity:\t");
  printvals(3,vel);
  sdvel(FARM,com,vel);
  printf("Fore Arm COM Velocity:\t");
  printvals(3,vel);
  sdvel(HARM,com,vel);
  printf("Hand COM Velocity:\t");
  printvals(3,vel);
  printf("\n");
*/
  printf("done.\n"); /* end of assembly & velocity analysis */

/* Kinematic Analysis. Loop calls sdmotion during each time step and stores
   the state vector in an array (neq x nsteps) until the desired duration
   has been achieved. sdmotion integrates the state vector over dt. */

  printf("Kinematics Analysis.....");

/*  fprintf(pfile_ptr,"%d\n",NSTEP); */

 fprintf(pfile_ptr,"Time\t\tAnkleAngle\tKneeAngle\tHipAngle\tShoulderAngle\tElbowAngle\tWristAngle");
```

```
fprintf(pmxfile_ptr,"Time\t\tAnklePmax\tKneePmax\tHipPmax\t\tShoulderPmax\tElbowPmax\tWristPmax
");


fprintf(pmnfile_ptr,"Time\t\tAnklePmin\tKneePmin\tHipPmin\t\tShoulderPmin\tElbowPmin\tWristPmin");

  fprintf(vfile_ptr,"Time\t\tAnkleVel\tKneeVel\t\tHipVel\t\tShoulderVel\tElbowVel\tWristVel");

  fprintf(afile_ptr,"Time\t\tAnkleAccel\tKneeAccel\tHipAccel\tShoulderAccel\tElbowAccel\tWristAccel");

  for (i = 0 ; i < NEQ ; i++) ystore[0][i] = y[i];
  for (i = 0 ; i < NEQ ; i++) dy[i] = 0.0;

  sdmotion(&t,y,dy,DT,CTOL,TOL,&flag,&err);
  if(err != 0){
    printf("\a\n Problem with integrator! Error no.: %d\n",err);
    exit();
  }

/*  for (i = 0 ; i < NQ ; i++) accel[0][i] = dy[NQ+i]; */
  accel[0][sdindx(LLEG,0)] = dy[NQ+sdindx(LLEG,0)];
  accel[0][sdindx(ULEG,0)] = dy[NQ+sdindx(ULEG,0)];
  accel[0][sdindx(TRUNK,0)] = dy[NQ+sdindx(TRUNK,0)];
  accel[0][sdindx(UARM,0)] = dy[NQ+sdindx(UARM,0)];
  accel[0][sdindx(FARM,0)] = dy[NQ+sdindx(FARM,0)];
  accel[0][sdindx(HARM,0)] = dy[NQ+sdindx(HARM,0)];
  accel[0][sdindx(HANDX,0)] = dy[NQ+sdindx(HANDX,0)];
  accel[0][sdindx(HANDY,0)] = dy[NQ+sdindx(HANDY,0)];
  accel[0][sdindx(HSLIDE,0)] = dy[NQ+sdindx(HSLIDE,0)];

  fprintf(pfile_ptr,"\n%f\t%f\t%f\t%f\t%f\t%f\t%f\t%f",t,y[sdindx(LLEG,0)],
        y[sdindx(ULEG,0)],y[sdindx(TRUNK,0)],y[sdindx(UARM,0)],
        y[sdindx(FARM,0)],y[sdindx(HARM,0)]);

/* Note that the angle LIMITS are in DEGREES, while other data is in radians.*/
  fprintf(pmnfile_ptr,"\n%f\t%f\t%f\t%f\t%f\t%f\t%f\t%f",t,ANKLEMIN,KNEEMIN,HIPMIN,
        SHLDRMIN,ELBOWMIN,WRISTMIN);
  fprintf(pmxfile_ptr,"\n%f\t%f\t%f\t%f\t%f\t%f\t%f\t%f",t,ANKLEMAX,KNEEMAX,HIPMAX,
        SHLDRMAX,ELBOWMAX,WRISTMAX);

  fprintf(vfile_ptr,"\n%f\t%f\t%f\t%f\t%f\t%f\t%f\t%f",t,y[NQ+sdindx(LLEG,0)],
        y[NQ+sdindx(ULEG,0)],y[NQ+sdindx(TRUNK,0)],y[NQ+sdindx(UARM,0)],
        y[NQ+sdindx(FARM,0)],y[NQ+sdindx(HARM,0)]);

  fprintf(afile_ptr,"\n%f\t%f\t%f\t%f\t%f\t%f\t%f\t%f",t,accel[0][sdindx(LLEG,0)],
        accel[0][sdindx(ULEG,0)],accel[0][sdindx(TRUNK,0)],
        accel[0][sdindx(UARM,0)],accel[0][sdindx(FARM,0)],
        accel[0][sdindx(HARM,0)]);

/* time & "state" array used to pass values to EVADS animation program. */

  time = t;
  for (i = 0 ; i < NQ ; i++) state[i] = y[i];

  flag = 0;
 /*
  printf("%f\t", t);
```

```
  printvals(NEQ,y);
*/
  for (i = 0 ; i < NEQ ; i++) dy[i] = 0.0;

  for(j = 1 ; j < NSTEP ; j++)
  {
    sdmotion(&t,y,dy,DT,CTOL,TOL,&flag,&err);
    if(err != 0){
       printf("\a\n Problem with integrator! Error no.: %d\n",err);
       exit();
     }

    for (i = 0 ; i < NEQ ; i++) ystore[j][i] = y[i];

/*    for (i = 0 ; i < NQ ; i++) accel[j][i] = dy[NQ+i]; */
    accel[j][sdindx(LLEG,0)] = dy[NQ+sdindx(LLEG,0)];
    accel[j][sdindx(ULEG,0)] = dy[NQ+sdindx(ULEG,0)];
    accel[j][sdindx(TRUNK,0)] = dy[NQ+sdindx(TRUNK,0)];
    accel[j][sdindx(UARM,0)] = dy[NQ+sdindx(UARM,0)];
    accel[j][sdindx(FARM,0)] = dy[NQ+sdindx(FARM,0)];
    accel[j][sdindx(HARM,0)] = dy[NQ+sdindx(HARM,0)];
    accel[j][sdindx(HANDX,0)] = dy[NQ+sdindx(HANDX,0)];
    accel[j][sdindx(HANDY,0)] = dy[NQ+sdindx(HANDY,0)];
    accel[j][sdindx(HSLIDE,0)] = dy[NQ+sdindx(HSLIDE,0)];

  fprintf(pfile_ptr,"\n%f\t%f\t%f\t%f\t%f\t%f\t%f",t,y[sdindx(LLEG,0)],
        y[sdindx(ULEG,0)],y[sdindx(TRUNK,0)],y[sdindx(UARM,0)],
        y[sdindx(FARM,0)],y[sdindx(HARM,0)]);

/* Note that the angle LIMITS are in DEGREES, while other data is in radians.*/

  fprintf(pmnfile_ptr,"\n%f\t%f\t%f\t%f\t%f\t%f\t%f",t,ANKLEMIN,KNEEMIN,HIPMIN,
        SHLDRMIN,ELBOWMIN,WRISTMIN);
  fprintf(pmxfile_ptr,"\n%f\t%f\t%f\t%f\t%f\t%f\t%f",t,ANKLEMAX,KNEEMAX,HIPMAX,
        SHLDRMAX,ELBOWMAX,WRISTMAX);

  fprintf(vfile_ptr,"\n%f\t%f\t%f\t%f\t%f\t%f\t%f",t,y[NQ+sdindx(LLEG,0)],
        y[NQ+sdindx(ULEG,0)],y[NQ+sdindx(TRUNK,0)],y[NQ+sdindx(UARM,0)],
        y[NQ+sdindx(FARM,0)],y[NQ+sdindx(HARM,0)]);

  fprintf(afile_ptr,"\n%f\t%f\t%f\t%f\t%f\t%f\t%f",t,accel[j][sdindx(LLEG,0)],
        accel[j][sdindx(ULEG,0)],accel[j][sdindx(TRUNK,0)],
        accel[j][sdindx(UARM,0)],accel[j][sdindx(FARM,0)],
        accel[j][sdindx(HARM,0)]);

/* Save values in external variables for access by EVADS animation prog. */

  time = t;
  for (i = 0 ; i < NQ ; i++) state[i] = y[i];

  }
/*
   printf("%f\t", t);
   printvals(NEQ,y);
*/
  printf("done.\n"); /* end of kinematics analysis */

 /* Inverse Dynamics Analysis */
```

```
  printf("Inverse Dynamics.....");

/* Calculation of joint torques from prescribed kinematics. */


fprintf(jtfile_ptr,"Time\t\tAnkleTorque\tKneeTorque\tHipTorque\tShoulderTorque\tElbowTorque\tWristTo
rque");


fprintf(tmnfile_ptr,"Time\t\tAnkleTmin\tKneeTmin\tHipTmin\t\tShoulderTmin\tElbowTmin\tWristTmin");


fprintf(tmxfile_ptr,"Time\t\tAnkleTmax\tKneeTmax\tHipTmax\t\tShoulderTmax\tElbowTmax\tWristTmax
");

  t=0.0;

/* Turn pescribed motion OFF for slider joints.*/

  sdpres(HANDX,0,0);
  sdpres(HANDY,0,0);
  sdpres(HSLIDE,0,0);

/* Turn prescribed motion ON for real body joints. */

  sdpres(LLEG,0,1);
  sdpres(ULEG,0,1);
  sdpres(TRUNK,0,1);
  sdpres(UARM,0,1);
  sdpres(FARM,0,1);
  sdpres(HARM,0,1);
/*  sdpres(HANDX,0,1);
  sdpres(HANDY,0,1);
  sdpres(HSLIDE,0,1); */

  for(j = 0 ; j < NSTEP ; j++){
    for (i = 0 ; i < NEQ ; i++) y[i] = ystore[j][i];

/*    sdmotion(&t,y,dy,DT,CTOL,TOL,&flag,&err);
    if(err != 0) printf("\a\n Problem with integrator! Error no.: %d\n",err);*/

  sdstate(t,&y[0],&y[NQ]);

  sdpresacc(LLEG,0,accel[j][sdindx(LLEG,0)]);
  sdpresacc(ULEG,0,accel[j][sdindx(ULEG,0)]);
  sdpresacc(TRUNK,0,accel[j][sdindx(TRUNK,0)]);
  sdpresacc(UARM,0,accel[j][sdindx(UARM,0)]);
  sdpresacc(FARM,0,accel[j][sdindx(FARM,0)]);
  sdpresacc(HARM,0,accel[j][sdindx(HARM,0)]);
  sdpresacc(HANDX,0,accel[j][sdindx(HANDX,0)]);
  sdpresacc(HANDY,0,accel[j][sdindx(HANDY,0)]);
  sdpresacc(HSLIDE,0,accel[j][sdindx(HSLIDE,0)]);

  sdderiv(dy,&dy[NQ]);

/* The function sdgetht determines the joint torques required to achieve
   the prescribed motion. */
```

```
  sdgetht(LLEG,0,&torq_lleg);
  sdgetht(ULEG,0,&torq_uleg);
  sdgetht(TRUNK,0,&torq_trunk);
  sdgetht(UARM,0,&torq_uarm);
  sdgetht(FARM,0,&torq_farm);
  sdgetht(HARM,0,&torq_harm);

/* Divide the joint torques by 2 to represent each arm and leg separately. */

  fprintf(jtfile_ptr,"\n%f\t%f\t%f\t%f\t%f\t%f\t%f",t,torq_lleg/2,torq_uleg/2,
        torq_trunk/2,torq_uarm/2,torq_farm/2,torq_harm/2);

/* Get torque max. and min. values from NASA joint strength software, or
   else assign estimated constant values. */

  torqlim(ind_table,y,tmn,tmx);

  fprintf(tmnfile_ptr,"\n%f\t%f\t%f\t%f\t%f\t%f\t%f\t%f",t,tmn[sdindx(LLEG,0)],
        tmn[sdindx(ULEG,0)],tmn[sdindx(TRUNK,0)],tmn[sdindx(UARM,0)],
        tmn[sdindx(FARM,0)],tmn[sdindx(HARM,0)]);

  fprintf(tmxfile_ptr,"\n%f\t%f\t%f\t%f\t%f\t%f\t%f\t%f",t,tmx[sdindx(LLEG,0)],
        tmx[sdindx(ULEG,0)],tmx[sdindx(TRUNK,0)],tmx[sdindx(UARM,0)],
        tmx[sdindx(FARM,0)],tmx[sdindx(HARM,0)]);

  t=t+DT;
  }
  printf("done.\n\n"); /* end of inverse dynamics */
}



/*----------------------------------------------------------------------*/
/*                    Functions                                         */
/*----------------------------------------------------------------------*/



/*************************************************************************

   sduforce

   This routine computes and applies the forces acting in the system. It must
   always be included, when the Simplified Analysis Routines are used, because
   they will make calls to sduforce. This is the case even if the sdforce
   function is unused (empty).

*************************************************************************/

sduforce(t,q,u)
double t,q[NQ],u[NU];
{
  double bdamp_harm,krot_harm,bias_harm,krot_harm_stop,tau_harm;

  double bdamp_trunk,krot_trunk,bias_trunk,krot_trunk_stop,tau_trunk;
  double bdamp_uleg,krot_uleg,bias_uleg,krot_uleg_stop,tau_uleg;
```

```
  double bdamp_lleg,krot_lleg,bias_lleg,krot_lleg_stop,tau_lleg;

/* WRIST */

  bdamp_harm = 2.0; /* N-m/(rad/sec) */
  krot_harm  = 18.14; /* N-m/rad */
  bias_harm  = -0.048; /* -0.048 = -2.75 deg (middle of range of motion) */
  krot_harm_stop = 1000.0; /* stiff spring for jointstops */

/* The passive wrist torques include a lower joint stop, a nominal range, and
   an upper joint stop. */

  if(q[sdindx(HARM,0)] < WRISTMIN*dtr)
    tau_harm = -krot_harm_stop*(q[sdindx(HARM,0)]-(WRISTMIN)*dtr);
  if(q[sdindx(HARM,0)] > WRISTMIN*dtr && q[sdindx(HARM,0)] < WRISTMAX*dtr)
    tau_harm = -krot_harm*(q[sdindx(HARM,0)]-bias_harm)
             -bdamp_harm*u[sdindx(HARM,0)];
  if(q[sdindx(HARM,0)] > WRISTMAX*dtr)
    tau_harm = -krot_harm_stop*(q[sdindx(HARM,0)]-(WRISTMAX)*dtr);

  sdhinget(HARM,0,tau_harm);

/* HIP */

  bdamp_trunk = 10.0; /* N-m/(rad/sec) */
  krot_trunk  = 100.0; /* N-m/rad */
  bias_trunk  = -52.0*dtr; /* desired position */
  krot_trunk_stop = 1000.0; /* stiff spring for jointstops */

/* The passive hip torques include a lower joint stop, a nominal range, and
   an upper joint stop. */
  if(q[sdindx(TRUNK,0)] < HIPMIN*dtr)
    tau_trunk = -krot_trunk_stop*(q[sdindx(TRUNK,0)]-(HIPMIN)*dtr);
  if(q[sdindx(TRUNK,0)] > HIPMIN*dtr && q[sdindx(TRUNK,0)] < HIPMAX*dtr)
    tau_trunk = -krot_trunk*(q[sdindx(TRUNK,0)]-bias_trunk)
             -bdamp_trunk*u[sdindx(TRUNK,0)];
  if(q[sdindx(TRUNK,0)] > HIPMAX*dtr)
    tau_trunk = -krot_trunk_stop*(q[sdindx(TRUNK,0)]-(HIPMAX)*dtr);

  sdhinget(TRUNK,0,tau_trunk);

/* KNEE */

  bdamp_uleg = 10.0; /* N-m/(rad/sec) */
  krot_uleg  = 100.0; /* N-m/rad */
  bias_uleg  = 47.0*dtr; /* desired position */
  krot_uleg_stop = 1000.0; /* stiff spring for jointstops */

/* The passive knee torques include a lower joint stop, a nominal range, and
   an upper joint stop. */

  if(q[sdindx(ULEG,0)] < KNEEMIN*dtr)
    tau_uleg = -krot_uleg_stop*(q[sdindx(ULEG,0)]-(KNEEMIN)*dtr);
  if(q[sdindx(ULEG,0)] > KNEEMIN*dtr && q[sdindx(ULEG,0)] < KNEEMAX*dtr)
    tau_uleg = -krot_uleg*(q[sdindx(ULEG,0)]-bias_uleg)
             -bdamp_uleg*u[sdindx(ULEG,0)];
  if(q[sdindx(ULEG,0)] > KNEEMAX*dtr)
    tau_uleg = -krot_uleg_stop*(q[sdindx(ULEG,0)]-(KNEEMAX)*dtr);
```

```
  sdhinget(ULEG,0,tau_uleg);

/* ANKLE */

  bdamp_lleg = 10.0; /* N-m/(rad/sec) */
  krot_lleg  = 100.0; /* N-m/rad */
  bias_lleg  = 21.0*dtr; /* desired position */
  krot_lleg_stop = 1000.0; /* stiff spring for jointstops */

/* The passive ankle torques include a lower joint stop, a nominal range, and
   an upper joint stop. */

  if(q[sdindx(LLEG,0)] < ANKLEMIN*dtr)
    tau_lleg = -krot_lleg_stop*(q[sdindx(LLEG,0)]-(ANKLEMIN)*dtr);
  if(q[sdindx(LLEG,0)] > ANKLEMIN*dtr && q[sdindx(LLEG,0)] < ANKLEMAX*dtr)
    tau_lleg = -krot_lleg*(q[sdindx(LLEG,0)]-bias_lleg)
             -bdamp_lleg*u[sdindx(LLEG,0)];
  if(q[sdindx(LLEG,0)] > ANKLEMAX*dtr)
    tau_lleg = -krot_lleg_stop*(q[sdindx(LLEG,0)]-(ANKLEMAX)*dtr);

  sdhinget(LLEG,0,tau_lleg);

}

/*****************************************************************************

  sdumotion

  If enabled, this function prescribes the motion of the hand.
  This function moves the endpoint of the arm along a desired trajectory (a
  circle with constant angular velocity) by means of prescribed motion in
  each of the two slider joints attached between the hand and ground.
  In addition, the hand is maintained in a horizontal orientation, by
  prescribing the angle of hslide wrt the slider to be 0.

*****************************************************************************/

sdumotion(t,q,u)
double t,q[NQ],u[NU];
{
    double xe, xde, xdde, ye, yde, ydde, theta;
    double puarm, vuarm, auarm, pfarm, vfarm, afarm;
    double pharm, vharm, aharm, qd[NQ], ud[NU];
    int presx=0, presy=0, presh=0, presuarm=0, presfarm=0, presharm=0;
    int presl=0, presu=0, prest=0;
    int i;
    extern double omega, rcirc, xstart, ystart;

    /* Kinematics */

    theta =  pi + omega * t;

    /* position */

    xe = xstart + rcirc * (1.0 + cos(theta));
    ye = ystart + rcirc * sin(theta);
```

```
  /* velocity */

  xde = omega * rcirc * (-sin(theta));
  yde = omega * rcirc * cos(theta);

  /* acceleration */

  xdde = omega * omega * rcirc * (-cos(theta));
  ydde = - omega * omega * rcirc * sin(theta);

  sdgetpres(HANDX,0,&presx);
  if (presx != 0){
    sdprespos(HANDX,0,xe);
    sdpresvel(HANDX,0,xde);
    sdpresacc(HANDX,0,xdde);
  }
  sdgetpres(HANDY,0,&presy);
  if (presy != 0){
    sdprespos(HANDY,0,ye);
    sdpresvel(HANDY,0,yde);
    sdpresacc(HANDY,0,ydde);
  }
  sdgetpres(HSLIDE,0,&presh);
  if (presh != 0){
    sdprespos(HSLIDE,0,-74.0*dtr);
    sdpresvel(HSLIDE,0,0.0);
    sdpresacc(HSLIDE,0,0.0);
  }

  sdgetpres(LLEG,0,&presl);
  if (presl != 0){
    sdprespos(LLEG,0,21.0*dtr);
    sdpresvel(LLEG,0,0.0);
    sdpresacc(LLEG,0,0.0);
  }
  sdgetpres(ULEG,0,&presu);
  if (presu != 0){
    sdprespos(ULEG,0,47.0*dtr);
    sdpresvel(ULEG,0,0.0);
    sdpresacc(ULEG,0,0.0);
  }
  sdgetpres(TRUNK,0,&prest);
  if (prest != 0){
    sdprespos(TRUNK,0,-52.0*dtr);
    sdpresvel(TRUNK,0,0.0);
    sdpresacc(TRUNK,0,0.0);
  }
}

/*****************************************************************************

  initconds

  This function sets up the initial conditions. The hand position is set to
  xstart, ystart values. This is referenced to the original position of the
  hand, with the arm hanging straight down. The x and y velocities of the
  hand are also set according to the value of omega (ang vel) value. Pass
  in y with an initial guess to control the assembly solution and improve
```

```
   convergence. On return, y should be a fully compatible state vector, unless
   an error is received.

*************************************************************************/

initconds(t,y)
double t, y[NEQ];
{

}

/*************************************************************************

   torqlim

   This function determines the joint torque limits, either by calling
   a NASA joint strength program, or by assigning constant values as
   estimates.

*************************************************************************/

torqlim(table,y,torqmin,torqmax)
double y[];
double torqmin[],torqmax[];
CoeffTable table[];
{
double angle;
double fptnm;
float t_coeff[COEFF];
float veloc;
int GetCoeff();

fptnm = 1.35582; /* Conversion of ft-lbs to N-m. */

/*************************************************************************
 Calls to GetCoeff(table,joint,axis,dir,veloc,coeff) in strengthlib_eva.c:

     table => ind_table (data structure, see strength_eva.h)
     joint => "right_shoulder"=0, "right_elbow"=1, "right_wrist"=2
     axis  => number sequentially for each axis available in that joint
              e.g. elbow="Y" only, but axis=0!!!! (ask NASA)
     dir   => (-1)=0, (+1)=1
     veloc => [deg/sec]
     coeff = returned array of coefficients in [ft-lbs]

   Note: angles used in regression eqn. must be in degrees.
*************************************************************************/

/* Ankle */

  torqmin[sdindx(LLEG,0)]=0.0; /* No data available yet */
  torqmax[sdindx(LLEG,0)]=0.0; /* No data available yet */

/* Knee */

  torqmin[sdindx(ULEG,0)]=0.0; /* No data available yet */
  torqmax[sdindx(ULEG,0)]=0.0; /* No data available yet */
```

```
/* Hip */

  torqmin[sdindx(TRUNK,0)]=0.0; /* No data available yet */
  torqmax[sdindx(TRUNK,0)]=0.0; /* No data available yet */

/* shoulder */

  angle=y[sdindx(UARM,0)]/dtr+61.5; /* Set 0 deg at -61.5 from ref. */
  if(angle < SHLDRMIN+61.5) angle=SHLDRMIN+61.5;
  if(angle > SHLDRMAX+61.5) angle=SHLDRMAX+61.5;
  if(angle < 0.)angle=angle*(-1.); /* absolute value of angle */
  veloc=y[NQ+sdindx(UARM,0)]/dtr;
  if(veloc < 0.)veloc=veloc*(-1.); /* absolute value of velocity */

    GetCoeff(table,0,1,0,veloc,t_coeff);
    torqmin[sdindx(UARM,0)]=-(t_coeff[0]+t_coeff[1]*angle
                    +t_coeff[2]*angle*angle)*fptnm;

    GetCoeff(table,0,1,1,veloc,t_coeff);
    torqmax[sdindx(UARM,0)]=(t_coeff[0]+t_coeff[1]*angle
                    +t_coeff[2]*angle*angle)*fptnm;


/* elbow */

  angle=y[sdindx(FARM,0)]/dtr;
  if(angle < ELBOWMIN) angle=ELBOWMIN;
  if(angle > ELBOWMAX) angle=ELBOWMAX;
  if(angle < 0.)angle=angle*(-1.);
  veloc=y[NQ+sdindx(FARM,0)]/dtr;
  if(veloc < 0.)veloc=veloc*(-1.);

    GetCoeff(table,1,0,0,veloc,t_coeff);
    torqmin[sdindx(FARM,0)]=-(t_coeff[0]+t_coeff[1]*angle
                    +t_coeff[2]*angle*angle)*fptnm;

    GetCoeff(table,1,0,1,veloc,t_coeff);
    torqmax[sdindx(FARM,0)]=(t_coeff[0]+t_coeff[1]*angle
                    +t_coeff[2]*angle*angle)*fptnm;


/* wrist */

  angle=y[sdindx(HARM,0)]/dtr;
  if(angle < WRISTMIN) angle=WRISTMIN;
  if(angle > WRISTMAX) angle=WRISTMAX;
  if(angle < 0.)angle=angle*(-1.); /* absolute value of angle */
  veloc=y[NQ+sdindx(HARM,0)]/dtr;
  if(veloc < 0.)veloc=veloc*(-1.);

    GetCoeff(table,2,0,0,veloc,t_coeff);
    torqmin[sdindx(HARM,0)]=-(t_coeff[0]+t_coeff[1]*angle
                    +t_coeff[2]*angle*angle)*fptnm;

    GetCoeff(table,2,0,1,veloc,t_coeff);
    torqmax[sdindx(HARM,0)]=(t_coeff[0]+t_coeff[1]*angle
                    +t_coeff[2]*angle*angle)*fptnm;
```

```
}

/*************************************************************************

   testconfig

   This function configures the system for the wrist, elbow, and shoulder
   tests. All joint angles are set to zero (wrt the reference configuration)
   except for the shoulder joint, which is set to +90 deg. Initial velocities
   are all set to 0.

*************************************************************************/

testconfig(t,y)
double t, y[NEQ];
{
  int i, lock[NU], fcnt, err;
  double yinit[NEQ];

  for (i = 0 ; i < NQ ; i++) sdpres(i,0,0);
  for (i = 0 ; i < NEQ ; i++) yinit[i] = 0.0; /* Also sets velocities = 0 */

  /* Set shoulder angle to 90 deg. */
  yinit[sdindx(UARM,0)] = 90.0*dtr;
  yinit[sdindx(HANDX,0)] = .63515;
  yinit[sdindx(HANDY,0)] = .63515;
  yinit[sdindx(HSLIDE,0)] = 90.*dtr;

  for (i = 0 ; i < NEQ ; i++) y[i] = yinit[i];

  for (i = 0 ; i < NU ; i++) lock[i] = 0;

/* Lock the position and velocity for handx, handy hslide, lleg, uleg, trunk
    and head to the passed-in values. */

  lock[HANDX] = 1;
  lock[HANDY] = 1;
  lock[HSLIDE] = 1;

  lock[LLEG] = 1;
  lock[ULEG] = 1;
  lock[TRUNK] = 1;

  sdassemble(t, y, lock, CTOL, 5000, &fcnt, &err);
  if (err != 0) printf("\a\n Assembly failed! Error no.: %d\n",err);

  for (i = 0 ; i < NU ; i++) lock[i] = 0;
  sdinitvel(t, y, lock, CTOL, 5000, &fcnt, &err);
  if (err != 0) printf("\a\n Velocity analysis failed! Error no.: %d\n",err);

 }

 /*************************************************************************

    wristtest

    This function tests the wrist torque. An accel of 5 deg/sec^2 is
```

```
   prescribed at the wrist (the rest being 0) and the required torque is
   determined.

*************************************************************************/

wristtest()
{
  int i;
  double t=0., y[NEQ], yinit[NEQ], dy[NEQ];
  double wristacc = 5.0, torq_harm;

  testconfig(t,&y);

  sdstate(t,&y[0],&y[NQ]);

  for (i = 0 ; i < NQ ; i++) sdpres(i,0,1);
  for (i = 0 ; i < NQ ; i++) sdpresacc(i,0,0.);

  sdpresacc(HARM,0,wristacc*dtr);
  sdpresacc(HANDY,0,0.0552*wristacc*dtr);
  sdpresacc(HSLIDE,0,wristacc*dtr);

  sdderiv(dy,&dy[NQ]);
  sdgetht(HARM,0,&torq_harm);
  printf("\n Wrist Test (%f deg/sec^2), torque = %f\n",wristacc,torq_harm);

}

/*************************************************************************

   elbowtest

   This function tests the elbow torque. An accel of 5 deg/sec^2 is
   prescribed at the elbow (the rest being 0) and the required torque is
   determined.

*************************************************************************/

elbowtest()
{
  int i;
  double t=0., y[NEQ], yinit[NEQ], dy[NEQ];
  double elbowacc = 5.0, torq_farm;

  testconfig(t,&y);
  sdstate(t,&y[0],&y[NQ]);

  for (i = 0 ; i < NQ ; i++) sdpres(i,0,1);
  for (i = 0 ; i < NQ ; i++) sdpresacc(i,0,0.);
  sdpresacc(FARM,0,elbowacc*dtr);
  sdpresacc(HANDY,0,0.3352*elbowacc*dtr);
  sdpresacc(HSLIDE,0,elbowacc*dtr);

  sdderiv(dy,&dy[NQ]);
  sdgetht(FARM,0,&torq_farm);
  printf("\n Elbow Test (%f deg/sec^2), torque = %f\n",elbowacc,torq_farm);

 }
```

```
/*****************************************************************************

   shouldertest

   This function tests the shoulder torque. An accel of 5 deg/sec^2 is
   prescribed at the shoulder (the rest being 0) and the required torque
   is determined.

*****************************************************************************/

shouldertest()
{
  int i;
  double t=0., y[NEQ], yinit[NEQ], dy[NEQ];
  double shoulderacc = 5.0, torq_uarm;

  testconfig(t,&y);
  sdstate(t,&y[0],&y[NQ]);

  for (i = 0 ; i < NQ ; i++) sdpres(i,0,1);
  for (i = 0 ; i < NQ ; i++) sdpresacc(i,0,0.);
  sdpresacc(UARM,0,shoulderacc*dtr);
  sdpresacc(HANDY,0,0.6352*shoulderacc*dtr);
  sdpresacc(HSLIDE,0,shoulderacc*dtr);

  sdderiv(dy,&dy[NQ]);
  sdgetht(UARM,0,&torq_uarm);
  printf("\n Shoulder Test (%f deg/sec^2), torque = %f\n\n",shoulderacc,torq_uarm);

}


/*****************************************************************************

   printvals

   Prints out an array of numbers.

*****************************************************************************/

void printvals(int nval, double array[])
{
  int i;
  for (i = 0 ;  i< nval ; i++) printf("%f\t", array[i]);
  printf("\n");
}
```